UNIVERSITÀ DEGLI STUDI DELL'AQUILA

DIPARTIMENTO DI INGEGNERIA E SCIENZE DELL'INFORMAZIONE E MATEMATICA

Dottorato di Ricerca in Ingegneria e scienze dell'Informazione

Curriculum Modelli computazionali emergenti: algoritmi, architetture software e sistemi intelligenti ciclo XXXV

Titolo della tesi

Models and algorithms with applications to wildfire management

SSD: ING-INF/05

Dottorando

Alessia Di Fonso

Coordinatore del corso

Prof. Vittorio Cortellessa

Tutor

Prof. Gabriele Di Stefano

Prof. Serafino Cicerone

a.a. 2021/2022

Contents

Summary

In the last twenty years, wildfires have grown in size and frequency. The risk of wildfires is mainly caused by the lack of proper fuel management, the abandonment of rural lands, and the growing number of people living in urban settlements in the proximity of wildland vegetation. The effect of a wildfire on the environment can be long-lasting; therefore it is important to identify and quantify the risks of a wildfire in a territory to adopt adequate preventive countermeasures. Choosing between alternative prevention strategies in a such complex scenario, need the integration of science and management. This thesis focuses on the design of models and algorithms useful in the context of fire preparedness measures and addresses the problem of the coordination of multi-robot systems for wildfire monitoring and intervention. We present a graph model able to evaluate the risk of fire in a territory considering the probabilities of fire ignition and propagation in a certain area. To prove the usability of the model, we applied it to a territory in the North of Corsica, an island of the Mediterranean area exposed to a high risk of wildfires due to dry and hot weather during summer. This case study constitutes a proof of concept showing how the model can be applied practically. The graph model incorporates data relevant to wildfire management from various sources like past fires and the result of software fire simulations. The outcome of the model is a risk cartography that associates the risk of fire to an area and localizes areas prone to a higher risk. Fire managers can then concentrate their budget and efforts to plan preventive strategies to reduce the risk in such areas. We also present a prototype of an easy to use web-application designed for fire and risk managers. It includes maps and built-in algorithms to help the planning and the evaluation of fire preventive strategies, like for example the installation of firebreaks. Firebreaks consist of a strip of land in which the fuel is removed. As a result, fire propagation beyond them is blocked or slowed down. Firebreaks are effective preparedness measures but they have a high economic and environmental impact, so their positioning must be planned. We define the firebreak location problem, to address the optimal positioning of firebreaks, study its complexity and present some cases solvable in polynomial time as well as heuristics. We then study algorithmic aspects of the coordination of multi-robotic systems. Robot technology has advanced quickly, assisting humans in increasingly complicated tasks. The development of robots able to operate in forest environments can help in tasks like firefighting and fire prevention by land monitoring. Robots

can monitor a land at high risk of fire by collecting pictures, videos, and acquiring data with various sensors, like, for example, smoke and thermal cameras. During a wildfire, multiple drones can cover vast areas and obtain supplementary views of a fire scene while reaching places that are not accessible or hazardous for humans. Autonomous, multi-robot systems can cooperate and self-organize providing a robust alternative to application-specific robots. However, the cooperation of multi-robotic systems is a challenging coordination task requiring algorithmic solutions. We address the problem from a theoretical point of view and present algorithms for the coordination of a group of robots. We solve different variations of the pattern formation problem. This is a fundamental problem in robotics in which robots must agree on their role and coordinate to place according to a given formation. Autonomous robots, able to arrange themselves into formation, could help in fire-extinguishing operations. Then, we introduce $\mathcal{MOBLOT}$, a new model for swarm and modular robotics, in which robots can cluster to create bigger computational units called molecular robots. $\mathcal{MOBLOT}$ allows us to model a swarm divided into sub-groups of robots that deploy and move in formation. These sub-groups can be employed for various goals, like patrolling, searching, and providing radio connections for firefighters. Adopting this new model, we study the solution to the pattern formation problem in a modular and hierarchical way, both in the Euclidean plane and on grid graphs.

Acknowledgments

The doctorate course has been a long learning journey allowing me to deepen my knowledge in a variety of fields. I would like to thank my supervisors Professor Gabriele Di Stefano and Professor Serafino Cicerone for their guidance, teachings, and support during the course and the suggestions and feedback they gave me during the writing of this thesis. I would also like to thank my supervisors and Professor Marc Demange, Professor Alfredo Navarra, and Professor Pierpaolo Vittorini for welcoming me into their research groups and for showing me how to structure and conduct scientific research.
On a personal level, I would like to thank my parents for being always there for me and for supporting my academic studies, and my sister, who is also currently working on her Ph.D. We shared the experience of being doctoral students, I would like to thank her for her moral support and her kindness.

Introduction

Wildfires are extreme phenomena that have a devastating impact on the environment and human activities. People's and animals' lives are in danger while the burnt vegetation takes decades to grow back. In the last years, wildfires are becoming more frequent and catastrophic, even in areas previously deemed as low risk. Recent vast wildfires registered in Chile (2017), Portugal (2017), Greece (2018), California (2017, 2018, 2020), Oregon (2020), Australia (2009, 2019-2020), Sweden (2014, 2018), Brazil (2020) and UK (2018) are only some examples. The possible causes include the decline of traditional rural systems, the reduction of resources allocated to forest management [138], the lack of proper wildfire protection measures [113], and the extreme weather events. These wildfires caused damage to social, economic, and environmental systems [135] pushing countries to increase their capacity to suppress fire [20] and to prepare specific security managers. Wildfire managers must take decisions quickly with incomplete knowledge of events, a limited budget, and huge amounts of data to analyze. Recent studies [106, 109] outline the need to improve the tools to support decision-making and effective planning of interventions. The coordinated action of firefighters and fire managers [106] is crucial to managing fire disasters effectively. Technologies such as mathematical models predicting fire spread and resource allocation tools help support efficient and accurate decision-making [106]. Fire management involves planning preparedness measures to mitigate the fire risk and deploying intervention actions to suppress a fire to protect people, cities, and natural resources. There is a growing necessity to integrate these two processes.

In this thesis, we propose mathematical models and computational algorithms to support managers in planning fire preparedness solutions and to support firefighters during the intervention phase. This work has been developed under the European project Geospatial-based Environment for Optimisation Systems Addressing Fire Emergencies (GEO-SAFE). The project built a research network connecting scientists from Europe, Australia, and firefighting agencies. Its main goals were the development of innovation in the efficient management of wildfires and the promotion of the exchange of knowledge among researchers and domain-based experts from a wide range of disciplinary fields, guaranteeing the best up-to-date methods. Among the objectives, a key one is to provide a way to measure the risk of fire in a

territory and to develop risk cartography. These require the development of models and efficient algorithms for fire prevention integrating preparedness activities with fire suppression actions. The models proposed under GEO-SAFE, apply probabilistic, combinatorial, and optimization methods to wildfire management. Optimization problems deal with simultaneous constraints on resources like, for example, the level of risk and budget, while efficient algorithms deal with theoretical limitations due to the hardness of problems. Therefore the study of specific cases typical of the context of wildfire management becomes relevant. Among the fire preventive actions, forest and land management are crucial to mitigate the spread of fire or to reduce the frequency of the incidents. In the first part of the thesis, we propose preparedness strategies to mitigate the risk of fire. We apply graph theory to model wildfire propagation in a territory and to measure the risk of fire in a territory. Graphs allow us to represent a landscape divided into areas while maintaining the adjacency between areas. The connectivity between areas helps in understanding the fire propagation paths and the planning of preparedness interventions to limit the damage of fires when they occur.

Successively, we focus on problems related to the construction of firefighting lines. A *firebreak* is a belt of land in which all the vegetation and organic matter are removed, to prevent or slow down fire from expanding towards populated areas and important buildings while providing access ways to firefighters [23, 114]. Firebreaks can be natural or constructed. Mountain crests, rivers, lakes, or creeks are a natural barrier to the spread of fire. Roads and cultivated fields can also work as firebreaks. The construction of firebreaks is a laborious and expensive activity. Firebreaks are typically between 3 and 20 meters broad, but their width depends on the type of fuel available on the site and the wind speed. In extreme cases, they can be even wider. Consequently, they have a high landscape and ecological impact and require periodic maintenance. To function properly, firebreaks require the vegetation level to be as low as possible [21, 23]. For all these reasons, it is impossible to create firefighting lines everywhere. So far, deciding the location of firebreaks remains an empirical process based on the statistical analysis of past fires and expertise practice. We formulate the Firebreak Location problem to address the optimal location of firebreaks within a territory to minimize a risk function under budget constraints. Due to the hardness of the problem, we study variations with different assumptions providing solutions for specific graph topologies and providing heuristic approaches. We validate the graph model on the landscape of Cap Corse, the peninsula in the North of Corsica island to prove the usability of the solution. We estimate the model parameters with the aid of software fire simulators. We visualize the results on geographical risk maps to give an intuitive and easy view of the areas with increased wildfire risk. We integrate maps and implemented algorithms in a prototype web application designed to support fire managers and decision-makers. The application allows end-users to visualize data about wildfire management and interact with built-in algorithms. Users can also simulate the installation of a fire preventive measure and see the effect in terms of risk reduction.

In the second part of the thesis, we focus on algorithms to coordinate a group of autonomous robots, like drones, that could be useful in wildfire management to support firefighters during the intervention phase and in monitoring tasks. Among the tasks that robots can accomplish, they can collect data [132] with various sensors like thermal cameras, smoke, humidity, or temperature sensors. For example, aerial robots could patrol a land at high risk of fire and collect detailed pictures or videos [82] to integrate information coming from satellites. Moreover, robots could be employed in fire monitoring [107] and provide information during fire suppression operations.

For these and other use cases, we present algorithms to place robots in a formation with given characteristics without the help of external supervision and a new model to move a group of robots in formation. This line of research follows from the studies on this topic presented during the GEO-SAFE project to solve the problem of gathering a group of robots in a chosen location [35, 41]. In case of a wildfire, a team of drones employed in firefighting operations could meet at the location where a fire broke out. While designing mobile robots, there are decisive aspects to consider, one of them being the algorithmic aspects of their coordination. This thesis focuses on models and algorithms to coordinate a group of autonomous robots that collaborate without centralized control. Each robot should have the minimal necessary capabilities needed to solve a task. The idea is to design a system that is robust to loss of memory, and lack of communication and resilient to temporary disruptions. Such robots might help in critical scenarios like a wildfire in which communication systems may be unavailable or compromised. The research of distributed agents, (often called robots) is twenty years long [73] and draws inspiration from the collective behavior shown by some animals and insects, like the flocking of birds or the spooling of fishes, the foraging of bees and ants. In these swarms, each group element has limited abilities compared to the complex behavior shown by the entire colony. Each animal or insect behaves according to local rules, while collective behavior emerges by interacting with the environment. The coordination of distributed robots is a challenging algorithmic task. We propose a solution for the arbitrary pattern formation problem in which robots, moving on grids, must be able to organize into a formation having any geometric shape assigned to them in input. As an example, robots could arrange themselves into formation to carry out fire-extinguishing operations or to monitor the land. Successively, we propose a distributed algorithm to solve the geodesic mutual visibility problem. This problem asks to place robots so that they are geodesic mutually visible: each couple of robots has a shortest path in which no other robot resides. The study is motivated by the fact that mutual visible robots can reach any other robot along a shortest path without collision. We present the first results achieved for robots disposed on the vertices of a tree. Then, we introduce $\mathcal{MOBLOT}$, a novel model in the context of theoretical swarm robotics in which robots can cluster to create bigger computational units called molecular robots inspired by the chemical paradigm in which atoms combine to make molecules. This model divides a set of robots into

subgroups, places each subgroup in formation, and moves the subgroup as a whole while staying in formation. These subgroups can cooperate to accomplish a shared goal or can be assigned different tasks. One possible application could be molecular robots supporting humans during a fire disaster. Robots could be divided into groups and employed for various goals, like patrolling, searching, and providing radio connections for firefighters. We present, under $\mathcal{MOBLOT}$ a variation of the pattern formation problem in which robots organize in formation hierarchically. We also apply the $\mathcal{MOBLOT}$ model for robots moving on grids that are often used in industrial applications for robot navigation.

List of Publications

The first part of the thesis, with the title **Fire Spread and risk mitigation**, is based on the joint work with Marc Demange, Gabriele Di Stefano, and Pierpaolo Vittorini. The results are presented in the papers:

[1] Marc Demange, Alessia Di Fonso, Gabriele Di Stefano, and Pierpaolo Vittorini. A graph theoretical approach to the firebreak locating problem. *Theoretical Computer Science*, 914:47–72, 2022.

[2] Marc Demange, Alessia Di Fonso, Gabriele Di Stefano, and Pierpaolo Vittorini. Network theory applied to preparedness problems in wildfire management. *Safety Science*, 152:105762, 2022.

[3] Marc Demange, Alessia Di Fonso, Gabriele Di Stefano, and Pierpaolo Vittorini. About the infinite windy firebreak location problem, submitted to journal, 2022.

The second part of the thesis, with the title **Models and algorithms for distributed robots**, is based on the joint work with Serafino Cicerone, Gabriele Di Stefano, and Alfredo Navarra. The results are presented in the papers:

[4] Serafino Cicerone, Alessia Di Fonso, Gabriele Di Stefano, and Alfredo Navarra. Arbitrary pattern formation on infinite regular tessellation graphs. In *Proceedings of the 22nd International Conference on Distributed Computing and Networking (ICDCN)*, pages 56–65. ACM, 2021.

[5] Serafino Cicerone, Alessia Di Fonso, Gabriele Di Stefano, and Alfredo Navarra. MOBLOT: molecular oblivious robots. In *20th International Conference on Autonomous Agents and Multiagent Systems, (AAMAS)*, pages 350–358. ACM, 2021.

[6] Serafino Cicerone, Alessia Di Fonso, Gabriele Di Stefano, and Alfredo Navarra. Molecular robots with chirality on grids. In *Algorithmics of Wireless Networks (ALGOSENSORS)*, pages 45–59. Springer International Publishing, 2022.

[7] Serafino Cicerone, Alessia Di Fonso, Gabriele Di Stefano, and Alfredo Navarra. Arbitrary pattern formation on infinite regular tessellation graphs. *Theoretical Computer Science*, 942:1–20, 2023.

[8] Serafino Cicerone, Alessia Di Fonso, Gabriele Di Stefano, and Alfredo Navarra. The geodesic mutual visibility problem for oblivious robots: The case of trees. In *Proceedings of the 24th International Conference on Distributed Computing and Networking (ICDCN)*, pages 150–159. ACM, 2023.

Part I

Fire spread and risk mitigation

Outline

Graphs are very abstract concepts that allow modeling a wide range of settings. They are also the primary tool to investigate the properties and structure of networks. Fundamental elements composing a graph are a set of elements, represented by vertices, and a set of connections represented by edges between the vertices. In a social network, vertices represent people while edges represent the interactions or friendships between them. In a communication network, vertices may represent computers while edges represent the possibility to exchange messages. We use graphs to represent a territory divided into areas. Each area corresponds to a vertex, while edges represent the adjacency between the areas and the possibility that a fire propagates to its neighbors. Fire spreading can be modeled as a propagation phenomenon over networks. These phenomena have been investigated for a long time in the fields of social science [80, 93] and epidemics [137]. Marketing studies the dynamics of diffusion of rumors in networks to model the adoption of a new product or an innovation in a community. In a network of contacts, the social interaction between friends and colleagues is decisive for the diffusion of innovations or ideas. If people trust their close contacts, they will likely believe their opinions. Information is said to go viral when it starts somewhere in a network and then quickly spreads on a global scale. Gifting a product to a subset of influential people in a social network can lead to a cascade effect in which friends recommend it to friends. One studied problem in viral marketing is how to choose the seeding set of influential people to maximize the diffusion of a product under budget constraints [93]. In the inverse problem, one may want to reduce the diffusion phenomenon in a network like the spreading of a disease in a population or of malicious software in a communication network, or in our case, fire in a territory. Two problems in these contexts are the CuttingEdge problem [95] and the Contamination Minimization problem [99]. They have very similar objectives to the Firebreak Location problem and consist in removing a fixed number of edges to minimize the risk. Tong et al. [133] consider removing a fixed number of edges to minimize the maximum eigenvalue in the final graph, given the assumption that the spreading risk relates to the largest eigenvalue of the adjacency graph. They demonstrate the NP-hardness of the problem and provide heuristics. Planarity is important in our case but, it is not considered in most studies aiming to control the spread of information. Indeed, the most natural case of fire spreading is to consider fire spreading from one area to

an adjacent area [2]: if each area is connected, then the resulting graph is planar. Modeling the landscape as a fire spread network is a classical approach to address fuel management problems (see, e.g., [31, 59, 69, 97, 105, 110, 120, 124]) or other fire emergency problems (see, e.g., [57, 58]). Russo et al. [124] model the terrain as a lattice where each node may burn, and fire propagation is a random walk from a starting vertex on fire. They use graph metrics like centrality statistics to identify vertices that contribute the most to fire spreading: these are the vertices where fuel reduction solutions should be applied. A few other combinatorial problems use a similar fire-spreading graph in a wildfire management context. The Fuel Management problem aims to schedule fuel reduction in the landscape to reduce the risk of spread [110]. Mahmoud et al. use an approach based on directed graphs and develop a decision tool to evaluate scenarios and to estimate the physical parameters that regulate the spread of fire [105]. Here we apply graph theory to model a landscape and apply models for diffusion phenomena to fire spread.

In Chapter 1, we introduce the graph model that allows describing the spread of fire. Then, we formulate the Firebreak Location problem to address the optimal location of firebreaks in a landscape to minimize a risk function under budget constraints. Successively, we study the complexity of the problem on planar graphs. We present an efficient polynomial time algorithm on tree topology. We present the Infinite Windy Firebreak Location problem, a variation of the Firebreak Location problem defined on infinite graphs, and study some cases solvable in polynomial time.

In Chapter 2 we apply heuristics methods useful for practical applications. We study a particular case of the Firebreak Location problem in which all the areas have the same probability to burn. We show that, when the probabilities of ignition are equal, the Firebreak Location problem can be reduced to the k-Graph Partition problem that consists in removing a fixed number of edges to split a graph in k connected components of balanced size. We test this technique on the geographical area of the North of Corsica. We validate the graph theoretical model by applying it to the landscape of Cap Corse a peninsula in the North Of Corsica. We describe how to compute the model quantities like the extension of the areas and estimate the probabilities of ignition for each area and the probabilities of spread for each edge. This case study proves the usability of the model. Then, we present a prototype web application designed to offer an effective view of data to target end-users, wildfire, risk managers, and fire agencies.

Chapter 1

The firebreak location problem

In this chapter, we propose a graph model to describe the spread of fire and to provide a way to compute fire risk within a territory. We formulate the Firebreak Location problem to address the optimal location of firebreaks in a landscape to minimize a risk function under budget constraints then we study the complexity of the problem on planar graphs. Due to the hardness of the problem, we look for cases solvable in polynomial time and study variations of the Firebreak Location problem on graph topologies like trees and grid graphs. We model the territory as a graph where vertices correspond to areas subject to fire with a certain ignition probability, and edges represent the possibility of fire spreading from one area to another. Directed edges model situations in which fire spreads in mainly one preferred direction, or in both with different probabilities because of dominant wind directions during the year. The construction of a firebreak is modeled as an edge cut that reduces the graph connectivity. Each edge has also associated a cost representing the cost of the edge removal. The Firebreak Location problem aims to identify the best positions for firebreaks that minimize the risk and respect budget constraints. Section 1.1 introduces the main notations (Subsection 1.1.1), the general problem (Subsection 1.1.2) and summarizes the related work (Subsection 1.1.3). Section 1.2 contains the main results concerning the computational complexity while Appendix A contains all the proofs, whereas Section 1.3 studies a polynomially solvable case on trees. In Section 1.4 we adapt the model for the Windy Firebreak Location problem, on infinite undirected graphs and introduce the Infinite Windy Firebreak Location problem. The land is modeled as an infinite graph and the goal is to find a cut system that allows the fire to be contained limiting the risk. Given an infinite graph, we assume that a fire ignites in a subset of vertices and propagates to the neighbors. The goal is to select a subset of edges to remove to contain the fire and avoid burning more than a finite part of the graph. Infinite graphs can be seen as a theoretical model of very large lands and then the problem is motivated by preventing a wildfire from escaping, i.e., becoming out of control. A conclusive discussion is then included in Section 1.5.

1.1 The model and the Firebreak Location Problem

In the following section we introduce the main notations, then in Section 1.1.2 we formalize the graph model and introduce the FIREBREAK LOCATION problem. The risk function and the firebreak location problem are related to similar approaches in the literature. In Section 1.1.3 we delve into the related works by comparing the risk function with the well-known *Independent Cascade model* [93] and the firebreak location problem with a selection of problems involving the removal of edges in a graph.

1.1.1 Main notations

In what follows we consider all graphs to be finite. Let $G = (V, E)$ be a directed graph (or digraph); a directed edge $e = \{x, y\} \in E$ between vertices $x, y \in V$ is denoted as $e = xy$.

A symmetric digraph is also called undirected graph where two edges xy, yx are replaced with a single undirected edge. When no ambiguity occurs, this undirected edge is denoted as xy (or yx). A mixed graph has directed and undirected edges. As a consequence, all graph notions not specifically restricted to directed or to undirected graphs are defined for mixed graphs and then are valid for directed and for undirected graphs as well. In a mixed graph, a path needs to respect edge orientation if it includes directed edges.

Given a mixed graph $G = (V, E)$ and two vertices $u, v \in V$, $u \xrightarrow{G} v$ means that there is a path from u to v in G (we say that v is *reachable* from u in G). For any set of vertices $V' \subset V$, $r_G(V') = \{v \in V, \exists u \in V', u \xrightarrow{G} v\}$ is the set of vertices reachable from V' in G.

By $G' \leq G$, we mean that $G' = (V, E'), E' \subset E$ is a partial graph of G. For any edge set $H \subset E$, we denote by G_H the partial graph $(V, E \setminus H)$ obtained from G by removing edges in H. Given a set $V' \subset V$, $G[V']$ denotes the subgraph induced by V' and any graph $G'' = (V'', E'')$, $V'' \subset V, E'' \subset E$ is called partial subgraph of G.

For any $k \geq 2$, $P_k = a_1 \dots a_k$ is a path on k vertices $a_1, \dots, a_k$ and edges $a_i a_{i+1}$, $i = 1, \dots k-1$. P_2 is a single edge. The extremities of the path are the two vertices of degree 1. Paths can be either undirected or directed from a_1 toward a_k. A cycle C_k is obtained from a P_k by adding an edge between its extremities; if it is directed, then the added edge is $a_k a_1$.

A mixed graph is called planar if it can be drawn in the 2-dimension plane without crossing edges. Such a drawing is called *planar embedding*. Given an undirected graph G, a subdivision of G is obtained by replacing edges uv by a path P_k, $k \geq 2$ (we add $k-2$ intermediate vertices). This transformation preserves planarity. K_p, $p \geq 1$ denotes an undirected clique on p vertices and $K_{p,q}$, $p, q \geq 1$, denotes a complete bipartite graph with respectively p and q vertices on each side. It is well-

known that $K_{3,3}$ is not planar and consequently, containing (as partial subgraph) a subdivision of $K_{3,3}$ is a certificate of non-planarity.

All graph-theoretical terms not defined here can be found in the book of Diestel [64]. For complexity concepts, we refer the reader to the work of Garey and Johnson [77].

1.1.2 Model and Problem formalization

We are given a mixed graph where vertices are subject to burn and edges represent potential fire spread from one vertex to an adjacent one. Fire ignitions may occur on some vertices. The objective is to select a set of edges, called *cut system*, to be blocked (removed) within a budget constraint to reduce the induced risk, as described below.

When an edge between two vertices is "treated" (blocked) (typically installing a firebreak corridor) the fire is blocked in both directions. For this reason, a treatment between vertices x and y is modeled as the removal of edges xy and yx, if they exist. To this end, we define a *cut system* as a subset H of E satisfying:

$$[(xy \in H) \wedge (yx \in E)] \Rightarrow (yx \in H).$$

An instance is defined by a mixed graph $G = (V, E)$, called (fire) *spreading graph*: every edge $e = xy \in E$ is assigned a probability of spread (in one direction if the edge is directed) $\pi_s(e)$ and a cost $\kappa(e)$ seen as the cost to cut xy and yx in case both edges exist. In this case, it is convenient in expressions to allocate half of the cost to xy and a half to yx, making κ just a symmetric function.

Every vertex $v \in V$ is assigned a value $\varphi(v)$ and a probability of ignition $\pi_i(v)$. The total value $\varphi(V')$ of a subset $V' \subset V$ is defined as $\varphi(V') = \sum_{v \in V'} \varphi(v)$. The cost of $E' \subset E$ is: $\kappa(E') = \sum_{e \in E'} \kappa(e)$.

Given a partial graph $G_S \leq G$ and a set $I \subset V$ of ignited vertices, the induced loss is:

$$\lambda(G_S, I) = \varphi(r_{G_S}(I)). \tag{1.1}$$

To evaluate the *risk* associated with H, we randomly select a set $I \subset V$ of vertices in the probability space $\mathcal{V} = (V, \pi_i)$ and a set $E_S \subset (E \setminus H)$ of edges in the probability space $\mathcal{E}_H = (E \setminus H, \pi_s)$, thus defining a random graph $G_S = (V, E_S) \leq G_H$. Both random choices are independent.

The probability of a set $I \subset V$ is:

$$\pi_i(I) = \prod_{v \in I} \pi_i(v) \times \prod_{v \notin I} (1 - \pi_i(v)). \tag{1.2}$$

Similarly, the probability of G_S is:

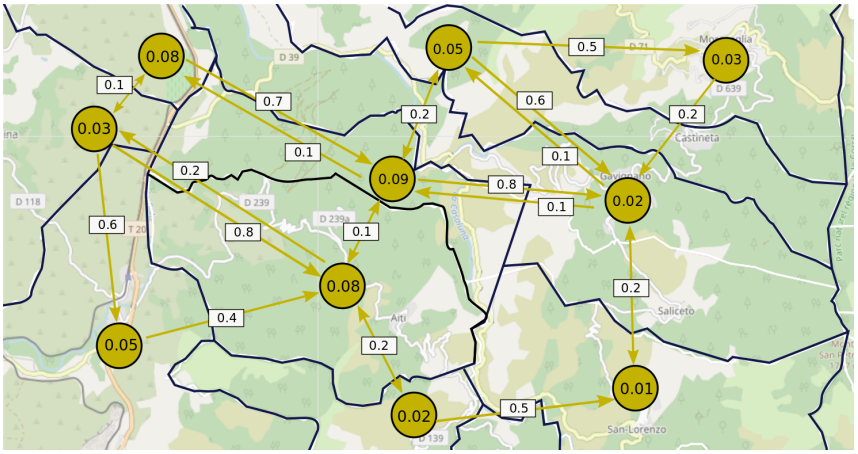


Figure 1.1: An example of a territory modeled with a graph. Nodes represent subareas with an ignition probability $\pi_i(v)$ while edges represent possible fire spread with an associated spreading probability $\pi_s(e)$.

$$\pi_s(G_S) = \prod_{e \in E_S} \pi_s(e) \times \prod_{e \notin E_S} (1 - \pi_s(e)) \tag{1.3}$$

Then, the risk associated with the cut system H is defined as follows:

$$\rho(G_H) = \sum_{I \subset V} \sum_{G_S \leq G_H} \pi_i(I)\pi_s(G_S)\lambda(G_S, I). \tag{1.4}$$

Definition 1.1.1. FIREBREAK LOCATION
Instance: a mixed graph $G = (V, E)$; for every edge $e \in E$ a probability of spread $\pi_s(e)$ and a cost $\kappa(e)$ (if xy and yx are in E, then $\kappa(xy) = \kappa(yx)$); for every vertex $v \in V$, a value $\varphi(v)$ and a probability of ignition $\pi_i(v)$. A total budget B and a total risk R.
Question: is there a cut system *$H \subset E$ such that $\kappa(H) \leq B$ and $\rho(G_H) \leq R$?*
We denote such an instance $(G, \pi_s, \pi_i, \kappa, \varphi, B, R)$.

From now, each time we mention a graph, it is a mixed graph unless differently stated. If we consider the optimization version instead of the decision version, then the threshold R is not part of the instance but becomes the objective to minimize. The particular case where all probabilities of spread are equal to 1 is called WINDY FIREBREAK LOCATION. Corresponds to considering that, without any intervention of firefighters, the fire will eventually spread. In this case, the definition of the problem can be simplified: we can directly define it on a mixed graph, stating that, two opposite edges xy and yx are always seen as an undirected edge. Then, the cost $\kappa(e)$ to remove an undirected edge $e = xy$ corresponds to the cost to remove xy and yx.

Definition 1.1.2. WINDY FIREBREAK LOCATION
Instance: a mixed graph $G = (V, E)$; for every edge $e \in E$ a probability of spread $\pi_s(e) = 1$ and a cost $\kappa(e)$; for every vertex $v \in V$, a value $\varphi(v)$ and a probability of ignition $\pi_i(v)$. A total budget B and a total risk R.
Question: is there a cut system *$H \subset E$ such that $\kappa(H) \leq B$ and $\rho(G_H) \leq R$?*
We denote such an instance $(G, \pi_s = 1, \pi_i, \kappa, \varphi, B, R)$.

Also, the definition of risk (Equation 1.4) can be simplified as stated in the following lemma. For a vertex $x \in V$ we denote by p_x the probability that x burns. For any cut system H, we denote by $U_{x,H} = \{t \in V, t \overset{G_H}{\rightarrow} x\}$ be the set of vertices t such that there is a path from t to x in G_H. Note that $x \in U_{x,H}$.

Proposition 1.1.3. *Let us consider an instance of* WINDY FIREBREAK LOCATION *on a graph G, a cut system H and a vertex x; then:*

$$\rho(G_H) = \sum_{x \in G_H} p_x \cdot \varphi(x), \text{ where } p_x = 1 - \prod_{t \in U_{x,H}} (1 - \pi_i(t)).$$

Proof. Using Equation 1.3 we have $\pi_s(G_H) = 1$ and $\forall\, G_S \leq G_H, G_S \neq G_H, \pi_s(G_S) = 0$. The expression of p_x is then a direct application of Equations 1.2 and 1.3 taking into account that the fire certainly spreads through connected components. Then, the expression of $\rho(G_H)$ is an immediate consequence of Equation 1.4 taking into account that, in the second sum, only the terms with $G_S = G_H$ are not null. □

The risk associated with a vertex x equals the probability that x burns multiplied by its value, $p_x \cdot \varphi(x)$. The total risk is the sum of the risks associated with each vertex. $\rho(G_H)$ cannot be computed in polynomial-time for general instances, as outlined in Section 1.1.3. As a consequence, FIREBREAK LOCATION is not necessarily in NP if $P \neq NP$. Proposition 1.1.3 implies that the objective value of WINDY FIREBREAK LOCATION can be computed in polynomial time. Let us remark that $\rho(G)$ is calculable in polynomial time when $k \log \Delta \in O(\log \log n)$, where k is the maximum length of a path between two vertices and Δ is the maximum vertex degree in the graph. Under these conditions, the number of nodes that can affect the probability of burning is $O(\Delta^k)$. The number of edges involved is therefore $O(\Delta^{2k})$ and a brute-force algorithm has to test $O(2^{\Delta^{2k}})$ realizations of the fire propagation on the subgraph, for each burning vertex of the graph. Imposing a polynomial time for this operation, we obtain $2^{\Delta^{2k}} \in O(n^t)$, for a constant t, and hence – applying two times the logarithmic operator – we have $k \log \Delta \in O(\log \log n)$.

1.1.3 Background and related works

Diffusion models. Here we report the fundamental characteristics of two diffusion models: the independent cascade and the linear threshold. The Independent Cascade model (IC) [93] describes the interaction between particle systems [67], then was investigated in the context of marketing [80], and was also used to model the spread of influence in social networks [93] or the spread of a disease [137]. Given a weighted graph $G = (V, E)$, the IC model is as follows: each edge $e = uv$ has a weight $p_e \in [0, 1]$ that is the probability that the edge e propagates an object (e.g., news, contagion) from vertex u to v. The diffusion process unfolds in discrete time steps. A vertex becomes active when it adopts an innovation, reacts to a viral marketing campaign or it gets infected. Each "active" vertex has a single chance to propagate the object to each of its susceptible neighbors with a probability of success p_{uv}. The attempts are independent random events. When an attempt succeeds, the targeted vertex becomes "active", then, at the next step, is considered for possible further propagation.

Linear Threshold (LT) model [130] is another paradigm to model the spread of information in a social network. Given a directed weighted graph $G = (V, E)$, each edge $e = uv$ has a weight $b_{u,v} \in [0, 1]$ and each vertex v has an activation threshold θ_v chosen in the interval $[0, 1]$ uniformly at random. The process unfolds in discrete time steps. An inactive vertex becomes active if the sum of the weights of its active incoming neighbors exceeds θ_v. The diffusion process ends when there are no new active vertices.

The *risk* $\rho(G_H)$ corresponds, in the IC model, to the propagation process modeled for the case of fire. Active vertices correspond, in our model, to be on fire. The primary difference with our model is the probability of ignition of vertices, while in the IC model, one vertex or a group of them is active at the beginning of the process. We could remove the probabilities of ignition by (i) introducing a vertex f (i.e., the fire) with $\varphi(f) = 0$, (ii) adding for each vertex $v \in V$ an edge $e = fv$ with $\pi_s(e) = \pi_i(v)$ and $\kappa(e) = \infty$, and then (iii) removing the probabilities of ignition. However, such a transformation does not necessarily preserve properties of the graph, like planarity, relevant while modeling a territory.

Computing the risk $\rho(G_H)$ is #P-hard in general graphs [30] and #P-complete in planar graphs [117], even with binary probabilities of ignition. Approximating $\rho(G_H)$ can be done as discussed by Kempe et al. in the unweighted case in [93], by providing an ε-approximation (approximation scheme) under some assumptions. To our knowledge, computing an approximated value in the general case is still open. Furthermore, Maehara et al. propose an algorithm that can be used to compute the exact value of $\rho(G_H)$ [104].

Two problems aimed to limit the diffusion of information in a network are the CuttingEdge problem [95] and the Contamination Minimization problem [98, 99]. Their objectives, similarly to Firebreak Location, consist in removing a fixed

number of edges to minimize the expected number of activated vertices. The former problem uses the LT model while the latter has been studied under both IC and LT models. A greedy approach guarantees a constant approximation ratio for the CuttingEdge problem [95]. Some greedy heuristics are considered for the Contamination Minimization problem [99] but, to our knowledge, no constant approximation is known. The main difference with Firebreak Location, excluding the diffusion model, is that the two problems are unweighted, with equal activation and there is no value associated with vertices or cost associated with edges. Moreover in Firebreak Location, the removal of one directed edge also removes the inverse edge if present, while both problems remove only directed edges. Blocking adversarial information in social networks is also studied in [90] as an interaction between a network defender whose aim is to limit the spread of misinformation, and an attacker whose goal is instead to maximize its diffusion. The problem is modeled as a Stackelberg game where the defender first chooses a set of nodes to block, and then the attacker selects a set of seeds to spread negative information from. Given the complexity of the problem, heuristic algorithms are applied. Another technique to contrast the spread of misinformation consists in placing monitors on the network, to detect misinformation [18, 142]. We conclude this section by mentioning a few problems in fire emergency management posed on the fire spreading graph and consequently, for which planar instances are relevant.

Russo et al. [124] model the terrain as a lattice (obtained by tessellation), where each vertex may burn. Centrality statistics are used to identify vertices in which deploy fuel reduction interventions. The difference with our model is twofold: first, firebreaks are placed on vertices, not on edges; this changes the combinatorial structure of the problem. Their model can be turned into ours by substituting every vertex x with two vertices x^+, x^- linked by one directed edge from x^+ to x^- in such a way that all other edges adjacent to x^+ (respectively, x^-) are entering (respectively, exiting) edges. The cost system could impose that only these edges can be part of a cut. On the other hand, in the undirected case, our model could be expressed as a vertex-based model in the line graph with vertex set as the edges of the original graph and edges between two vertices representing adjacent edges. However, in this case, the spread of fire in the original graph cannot be easily represented by the spread in the obtained graph. Secondly, the objective function is not directly linked to the risk we use in this work.

A few other combinatorial problems use a similar fire-spreading graph in a wildfire management context. The Fuel Management problem aims to schedule fuel reduction [110]. The instance is similar to Firebreak Location; the main difference is that treating a zone (prescribed burning or harvesting) corresponds to removing all edges adjacent to the related vertex, which makes this problem vertex-based. A second difference is that a vertex, removed from the graph, may reappear after a few years with the natural growth of the vegetation. The objective is to schedule preventive treatments over a long period. To our knowledge, most approaches to

solving Fuel Management use integer linear formulations for which it is hard to exploit planarity. Nevertheless, Demange and Tanasescu consider a multi-period Fuel Management problem from a graph optimization perspective [59]. The problem is reduced to a Vertex Cover problem and is shown to be NP-hard on planar graphs. Planarity is then exploited to derive an asymptotic polynomial approximation scheme.

1.2 Complexity results for planar instances

Many different hardness's notions have been developed depending on the kind of problems: the most popular notions are *NP-completeness* and *NP-hardness* fully explained in [77].

The majority of NP-hardness results are for general instances; however, an accurate evaluation of the computational complexity of the considered model generally demands more restrictive hardness results for restricted classes of instances relevant to the application. The challenge is then to show to which extent the problem restricted to these classes of instances is still hard. For a hardness result, the more restricted the class of instances the stronger the result. General hardness results may not be relevant if the instances emerging from the application have strong structural properties that can be used to solve the model efficiently.

In Section 2.2 we show that, when all probabilities of ignition are equal, Windy Firebreak Location in an undirected graph can be reduced to the k-Graph Partition problem that consists in removing a fixed number of edges to split a graph in k connected components of balanced size. This argument allows us to prove the NP-hardness of Windy Firebreak Location in any class of graph where k-Graph Partition is NP-hard and in particular in unit-disk graphs [68]. To our knowledge, the complexity of k-Graph Partition is not known in restricted classes of planar graphs that are the most natural case for our problem in the context of wildfire management. In our case, the graph represents the adjacency of zones in the considered territory, and consequently, the case of planar graphs and even with very low vertex degrees seems particularly relevant. As said before, $\rho(G)$ is computable in polynomial time and it is possible to show that the problem is in NP. The general case is much harder: computing $\rho(G)$ has been shown #P-hard [30] and #P-complete in planar graphs [117].

We investigate the complexity of Windy Firebreak Location in restricted planar instances that are natural in a real context. For completeness, we report the reduction in Appendix A as published in [1]. Table 1.1 summarizes all the complexity results for finite graphs. We first study a restricted version of Planar Max 2SAT that is used for our main reduction. We define the problem and prove its NP-completeness. Then, we prove that Windy Firebreak Location is hard on stars if edge costs and vertex values can be any integer.

Firebreak Location									
Graph (V, E)	Δ	π_s	π_i	κ	φ	B	R	*Complexity*	*Reference*
general								NP-hard	Prop. 1.2.1

Windy Firebreak Location									
Graph (V, E)	Δ	π_s	π_i	κ	φ	B	R	*Complexity*	*Reference*
unit-disk		1	uniform		1			NP-C	[2]
star		1	0/1					NP-C	Prop. 1.2.1
bipartite planar	4	1		1	1			NP-C	Th. 1.2.2
subgrid	4	1		1	0/1			NP-C	Prop. 1.2.3
grid	4	1		0/1	0/1			NP-C	Prop. 1.2.3
trees (Algo. 1)		1	0/1	1	1	poly	opt	$O(\|V\| \cdot B^2)$	Th. 1.3.1
trees		1	0/1	$\mathbb{N}$		poly	opt	$O(\|V\| \cdot B^2)$	Th. 1.3.2

Table 1.1: Results presented for finite graphs— Note: *poly* stands for $O(Poly(|V|))$, *opt* states that we find the minimum risk value, an empty cell means that any value is acceptable.

Proposition 1.2.1. Partition *polynomially reduces to* Windy Firebreak Location *on stars.*

Successively, we show that Windy Firebreak Location is NP-complete in bipartite planar graphs of degree at most 5 in the polynomially bounded case, i.e., with vertex values and edge costs bounded by a polynomial function. We use a reduction from the restricted version of Planar Max 2SAT. Finally, we use self-refinements to show that the problem remains NP-complete in bipartite planar graphs of degree at most 4 and with all values of vertices and costs of edges equal to 1. Here we report the main results.

Theorem 1.2.2. Windy Firebreak Location *is NP-complete in bipartite planar graphs of maximum degree 4 with all vertex values and edge costs equal to 1.*

Proposition 1.2.3. Windy Firebreak Location *is NP-complete if:*

- *the graph is a subgrid with binary vertex values and unitary edge costs;*
- *the graph is a grid with binary vertex values and edge costs.*

It is worth noting that the results concerning the bound on the degrees are relevant when the territory is divided into adjacent areas forming a grid. In this case, the underlying graph has vertex degrees at most four.

In the next section, we identify a case solvable in polynomial time for the WINDY FIREBREAK LOCATION problem.

1.3 Windy Firebreak location in Trees

The hardness results in the previous section motivate the question of identifying some cases solvable in polynomial time for Firebreak Location. In particular, as seen in Proposition 1.2.1, Windy Firebreak Location is hard on trees (even on stars) if edge costs and vertex values can be any integer. Since Windy Firebreak Location revealed to be hard in restricted cases and since its complexity with binary probabilities of ignition is still open, it seems to us relevant to start with this case. In this section, we present a polynomial algorithm that solves Windy Firebreak Location with all edge costs and vertex values equal to 1 in a general tree with a subset of burning vertices. The algorithm outputs the maximum number of vertices that can be saved and the corresponding cut system.

Given a tree $T = (V, E)$, we consider an instance $(T, \pi_s, \pi_i, \kappa, \varphi, B)$ of the Firebreak Location problem (optimization version) where:

- $\pi_s(e) = 1$, for each $e \in E$ (this is an instance of Windy Firebreak Location);
- $\pi_i(v) = 1$, for $v \in V' \subseteq V$; and $\pi_i(v) = 0$ for $v \in V \setminus V'$;
- $\kappa(e) = 1$, for each $e \in E$;
- $\varphi(v) = 1$, for each $v \in V$;
- a given budget B.

We devise a polynomial time algorithm that computes a *cut system* $H \subset E$ such that $\kappa(H) \leq B$ minimizing the risk, which is equivalent to maximizing the number of saved vertices.

Given an instance $(T, \pi_s = \mathbb{1}, \pi_i, \kappa, \varphi, B)$ of the optimization version of Windy Firebreak Location, where T is a tree, we choose a vertex as the root, and orient edges from the root to the leaves. For every vertex, we define an order of its children and then we number the vertices $v_0, \ldots, v_{|V|-1}$ in post order. Let T_i be the subtree rooted in the vertex v_i that includes only vertices that are descendant of v_i in T. By property of the post order, if T_i is a subtree of T_j, then $i < j$. Given a cut system H, we denote with $\pi_o(v)$ the probability, in G_H, that a vertex burns in the final setting. $\pi_o(v) = 0$ if the vertex v does not burn in the solution or $\pi_o(v) = 1$ if the vertex v burns in the final setting.

Input: A tree T, a set of vertices $V' \subset V$ such that $\pi_i(v) = 1$ for $v \in V'$, $\pi_i(v) = 0$ for $v \notin V'$ and a budget B.

Output: A *cut system* $H \subset E$ of cost at most the given budget B and maximizing the number of vertices v such that $\pi_o(v) = 0$.

1.3.1 Algorithm description

The algorithm computes an optimal solution using two nested dynamic programming processes. The main dynamic programming process computes an optimal solution for each subtree starting from the leaves and following a post-order visit. At each step, the algorithm computes an optimal solution for a subtree T_i using solutions for the subtrees induced by the children of v_i (already computed due to the post-order visit), using a second dynamic programming process. The procedure continues until an optimal solution is computed for the whole tree.

In a more formal way, given the tree $T = (V, E)$ and a vertex r as the root, the algorithm visits and numbers the vertices in post-order from v_0 to $v_{|V|-1}$ ($v_{|V|-1} \equiv r$). Given a vertex v_i, we denote with v_{i_j} the j-th children of v_i (in the chosen ordering of children of each vertex) and, by definition of the post-order, if $j < j'$, then $i_j < i_{j'} < i$.

The algorithm then builds two tables (see Figure 1.2). For both tables, rows and columns are numbered starting from zero.

Table A

Table A (main dynamic programming process) has $|V|$ rows and $B + 1$ columns. It contains, for each row i and each budget $b \in \{0, \ldots, B\}$, the number of vertices of T_i that can be saved for two different scenarios - if v_i burns and if v_i does not burn - in the final setting, and, for each case, a corresponding cut system of b edges in T_i. If $v_i \in V'$, then only the case where v_i burns is taken into account. So, every entry in the table is a 4-tuple $A_{i,b} = (f^+, f^-, H^+, H^-)$ related to the subtree T_i, rooted in v_i and for a given budget b. f^+ is the optimal value when vertex v_i burns (i.e., $\pi_o(v_i) = 1$), f^- is the optimal value when vertex v_i is not burning in the final setting (i.e., $\pi_o(v_i) = 0$), H^+ and H^- are optimal cut systems associated with f^+ and f^-, respectively. In what follows, we respectively denote by A_{i,b,f^+}, A_{i,b,f^-}, A_{i,b,H^+}, and A_{i,b,H^-} the four components of $A_{i,b}$.

Algorithm 1 builds Table A. It is straightforward: a root is chosen at line 2, a post order visit is executed at line 3, each row of Table A is filled using procedure TableST (see Algorithm 2 described below) at lines 4–5. Once the values for Table A are computed for all the subtrees T_i, the element in the last row and column contains an optimal solution for both possible states of the root r (i.e., burns or does not burn).

An optimal solution for the problem is then the maximum between these two values, with the corresponding cut system. It is returned at lines 6–9.

Algorithm 1 Optimal Tree Cut

Input: instance $I = (T, \mathbb{1}, \pi_i, \kappa, \varphi, B)$ of WINDY FIREBREAK LOCATION with $\pi_i(v) \in \{1, 0\}$.

Output: (optimal cut system H, the number of saved vertices)

1: **procedure** TABLEA(I)
2: pick a vertex r as the root
3: let $v_0, v_1, \ldots, r = v_{|V|-1}$ the vertices of T visited in post-order from r
4: **for** $i = 0, 1, \ldots, |V| - 1$ **do**
5: $A_i \leftarrow$ TABLEST(T,A,v_i,B)
6: **IF** $A_{|V|-1,B,f^-} > A_{|V|-1,B,f^+}$ **THEN**
7: **RETURN** $(A_{|V|-1,B,H^-}, A_{|V|-1,B,f^-})$
8: **ELSE**
9: **RETURN** $(A_{|V|-1,B,H^+}, A_{|V|-1,B,f^+})$

Table ST

Table ST (auxiliary dynamic programming process) is built for each subtree T_i rooted in v_i to compute the i-th row of Table A using row $i-1$. Each row of Table ST stores, for each possible budget $b \in \{0, \ldots, B\}$, solutions (value and cut system) for some subtrees of T_i if v_i burns and, when $v_i \notin V'$, if it does not burn. Table ST has $k+1$ rows, where k is the number of children of v_i, and $B+1$ columns. The first row stores solutions (optimal value and a related cut system) for the root v_i without descendants (so, a single vertex). If $k > 0$, then each row $j > 0$ is filled with solutions for the subtree T_i^j obtained by connecting to v_i the subtrees $T_{i_1}, \ldots, T_{i_j}$. The last row then corresponds to T_i. Each column b of Table ST corresponds to the budget used for the related subtree. Note that the last row of Table ST for vertex i is the i-th row of Table A. Like in Table A, every entry in Table ST is a 4-tuple $ST_{j,b} = (f^+, f^-, H^+, H^-)$, for subtree T_i^j and budget b. We use the same abbreviated notation as for Table A, respectively denoting ST_{j,b,f^+}, ST_{j,b,f^-}, ST_{j,b,H^+}, and ST_{j,b,H^-} the four components of $ST_{j,b}$.

The values of row $j > 0$ of Table ST for subtree T_i are computed as follows. The root v_i of subtree T_i can either burn or not if $v_i \notin V'$ and certainly burns if $v_i \in V'$, and the same occurs for vertex v_{i_j}, root of T_{i_j}. So, when completing the j-th row of Table ST for T_i, we have at most four possible combinations. The case where a vertex does not burn is only considered if this vertex has a probability of ignition 0 (i.e., it is not in V'). Two of these cases are concordant (i.e, v_i and v_{i_j} both burn or both do not burn), then no cut is needed between them, and two of them are discordant (one of v_i and v_{i_j} is burning and one is not) so the edge connecting them must be cut in any feasible solution. Note that $-\infty$ is used as value to state that there is no related feasible solution. This is the case when v_i is in V' and is stated not burning in the combination under analysis.

More precisely, Algorithm 2 implements the procedure for Table ST. For each budget $b \in \{0, \ldots, B\}$, the first row is filled with values $(0, 1, \emptyset, \emptyset)$ if $\pi_i(v_i) = 0$ (i.e., v_i is not burning) and $(0, -\infty, \emptyset, \emptyset)$ if $\pi_i(v_i) = 1$ (i.e., v_i is burning).

If v_i is not a leaf, then the subsequent rows are computed considering the edge between vertex v_i and the latest added vertex v_{i_j}. To compute ST_{j,b,f^+} (v_i burns in the final setting), for budget b, we consider the latest added vertex v_{i_j} and distinguishing the two possible cases whether v_{i_j} burns or not.

1. *both v_i and v_{i_j} burn.* The algorithm finds the maximum value obtained by allocating x and $b - x$ budget on T_i^{j-1} and T_{i_j} for $x \in \{0 \ldots b\}$ and summing the values of ST_{j-1,b,f^+} with A_{i_j,b,f^+} (lines 9–11);

2. *v_i burns and v_{i_j} does not burn.* In any feasible solution, we must cut the edge $v_i v_{i_j}$ and allocate the remaining $b - 1$ budget on the two subtrees. So, we distribute x and $b - x - 1$ budget on the two subtrees and pick the maximum values between the sums of ST_{j-1,b,f^+} with $A_{i_j,b-1-x,f^-}$ (lines 13–14). If the value computed in this case is greater than the value computed in the previous one, then the solution is updated and edge $v_i v_{i_j}$ is added to the cut-system H^+ (lines 15–17).

The best value obtained in cases 1 and 2 corresponds to the correct value for ST_{j,b,f^+} assuming that values of ST_{j-1,b,f^+}, A_{i_j,b,f^+} and A_{i_j,b,f^-} are correct.

To compute ST_{j,b,f^-} now, the algorithm evaluates the two cases in which v_i does not burn in the final setting and v_{i_j} either burns or not. This case is similar to the previous one with the possible outputs computed at lines 20–22 for the discordant case, lines 23–28 for the concordant one. When table ST is completed, the last row contains the values of the solution for subtree T_i, for all varying budgets (line 29).

Figure 1.2 shows an example of a tree with eight vertices, two of them burning (vertices 3 and 7). For simplicity, the tables do not include the related cut systems, but only the number of saved vertices. The top table depicts the solutions computed for T_0, for each possible budget, as stored in Table ST. The values 0 and 1 in "0/1" correspond to the number of vertices saved when vertex v_0 burns and does not burn, respectively. Note that these values are not affected by the budget because they refer to a leaf of T (and consequently, the related subtree has no edge). These values fill the first row of Table A. The center of Figure 1.2 shows Table ST for the subtree T_3: all possible subtrees to inspect are in the first column, and the solutions – for all budget values – are in the subsequent columns. When Table ST is complete, its last row represents an optimal solution for T_i^k, i.e., the whole subtree T_i, for the possible states of v_i, and then this row is copied into Table A at row i. Finally, the bottom of Figure 1.2 corresponds to Table A, which shows that the optimal value is the maximum between 4 and $-\infty$. The value 4 then states that four vertices can be saved by applying the associated cut system (i.e., that cuts the edges $v_3 v_0$, $v_3 v_1$, and $v_7 v_6$).

Algorithm 2 Procedure for table ST

Input: Tree T, table A, vertex v_i and budget B
Output: optimal solutions for the subtree T_i, for each budget b up to B

procedure TABLEST(T,A,v_i,B)
  let $(v_{i_1}, v_{i_2}, \ldots, v_{i_k})$ be the children of vertex v_i
  if $\pi_i(v_i)$=1 **then**
    $ST_{0,b} \leftarrow (0, -\infty, \emptyset, \emptyset) \qquad \forall\ b \in \{0, 1, \ldots, B\}$
  else
    $ST_{0,b} \leftarrow (0, 1, \emptyset, \emptyset) \qquad \forall\ b \in \{0, 1, \ldots, B\}$
  for $j = 1, 2, \ldots, k$ **do**
    for $b = 0, 1, \ldots, B$ **do**
      $z \leftarrow \arg\max_{x \in \{0,\ldots,b\}} \{ST_{j-1,x,f^+} + A_{i_j,b-x,f^+}\}$ ▷ v_{i_j} and v_i both burn
      $ST_{j,b,f^+} \leftarrow ST_{j-1,z,f^+} + A_{i_j,b-z,f^+}$
      $ST_{j,b,H^+} \leftarrow ST_{j-1,z,H^+} \cup A_{i_j,b-z,H^+}$
      if $(\pi_i(v_{i_j}) = 0) \wedge (b \geq 1)$ **then** ▷ v_i burns, v_{i_j} not, budget ≥ 1
        $z' \leftarrow \arg\max_{x \in \{0,\ldots,b-1\}} \{ST_{j-1,x,f^+} + A_{i_j,b-1-x,f^-}\}$
        $m' \leftarrow ST_{j-1,z',f^+} + A_{i_j,b-1-z',f^-}$
        if $m' \geq ST_{j,b,f^+}$ **then**
          $ST_{j,b,f^+} \leftarrow m'$
          $ST_{j,b,H^+} \leftarrow ST_{j-1,z',H^+} \cup A_{i_j,b-1-z',H^-} \cup \{(v_i, v_{i_j})\}$
      if $\pi_i(v_i) = 0$ **then** ▷ v_i does not burn
        if $b \geq 1$ **then** ▷ v_{i_j} burns, budget ≥ 1
          $z \leftarrow \arg\max_{x \in (0,\ldots,b)} \{ST_{j-1,x,f^-} + A_{i_j,b-1-x,f^+}\}$
          $ST_{j,b,f^-} \leftarrow ST_{j-1,z,f^-} + A_{i_j,b-1-z,f^+}$
          $ST_{j,b,H^-} \leftarrow ST_{j-1,z,H^+} \cup A_{i_j,b-1-z,H^+} \cup \{(v_i, v_{i_j})\}$
        if $\pi_i(v_{i_j}) = 0$ **then** ▷ v_{i_j} does not burn
          $z' \leftarrow \arg\max_{x \in (0,\ldots,b)} \{ST_{j-1,x,f^-} + A_{i_j,b-x,f^-}\}$
          $m' \leftarrow ST_{j-1,z',f^-} + A_{i_j,b-z',f^-}$
          if $m' \geq ST_{j,b,f^-}$ **then**
            $ST_{j,b,f^-} \leftarrow m'$
            $ST_{j,b,H^-} \leftarrow ST_{i-1,z',H^-} \cup A_{i_j,b-z,H^-}$
  return ST_k

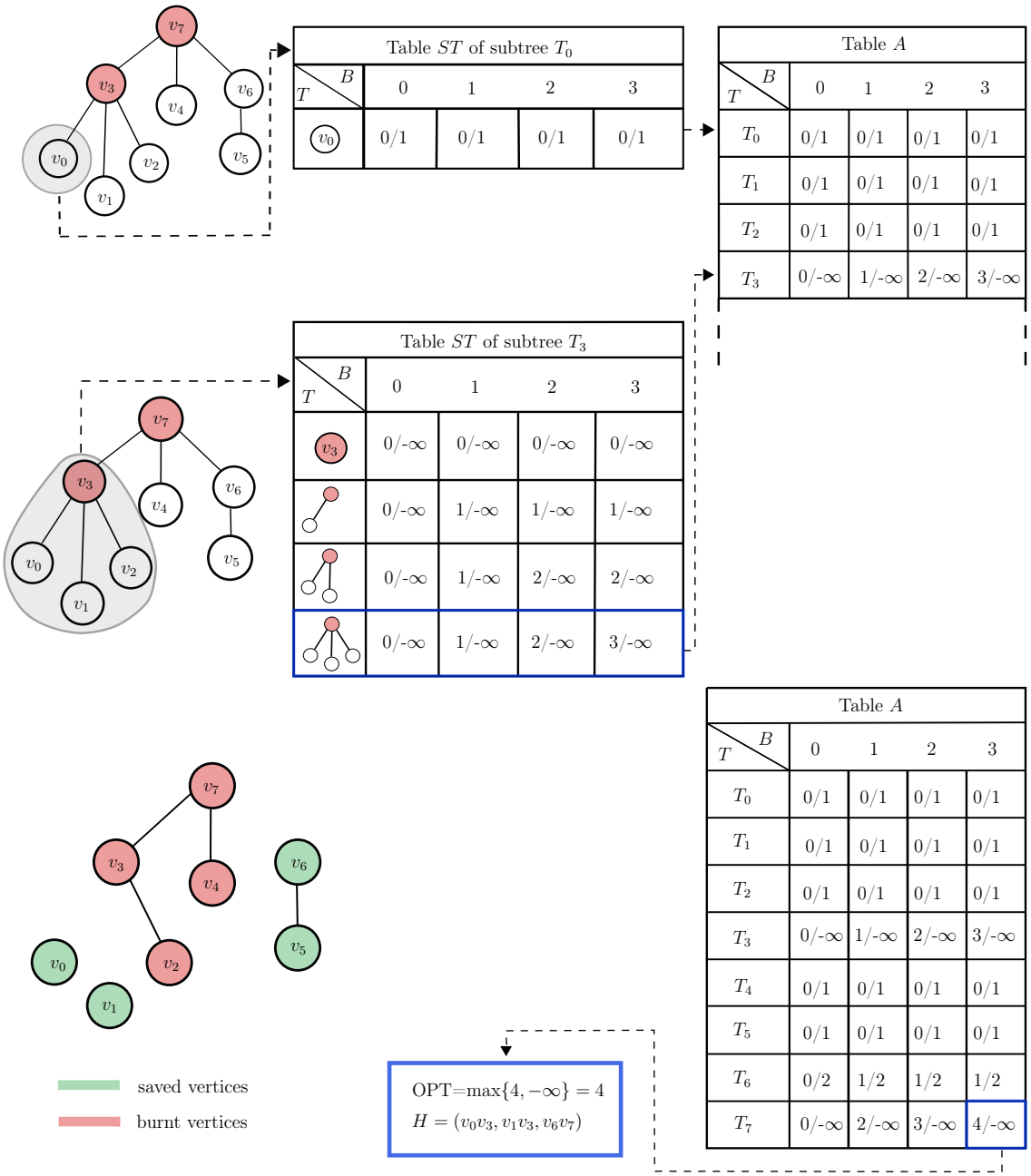


Figure 1.2: Different phases of the computation of an optimal *cut system* H for a tree T with $\pi_i(3) = \pi_i(7) = 1$ and a budget B=3. At the top, the figure shows how to compute Table A from Table ST. Table ST is shown for subtrees T_0 and T_3. At the bottom right, the figure shows the completed Table A. The optimum solution, that is the maximum number of vertices that can be saved from fire, is found in the last row and column of Table A. The optimum *cut system* H is computed by Algorithm 1 but is not represented in the picture. At the bottom left, the figure shows the tree T in which the edges from the cut system H are removed.

1.3.2 Algorithm correctness and computational complexity

We complete the section with two cases solvable in polynomial time on trees. First, Theorem 1.3.1 directly uses Algorithm 1. Then, we sketch how to generalize it to the case of any vertex value and polynomially bounded integral edge costs, which leads to our last result, Theorem 1.3.2.

Theorem 1.3.1. *Algorithm 1 correctly computes an optimal solution for an instance* $I = (T, \mathbb{1}, \pi_i, \mathbb{1}, \mathbb{1}, B)$ *of* Windy Firebreak Location *where* $T = (V, E)$ *is a tree and* $\pi_i(v) \in \{1, 0\}$. *The computational time is* $O(|V| \cdot B^2)$.

Proof. We prove the correctness by induction on the rank in post order. For each leaf v_i of T, Algorithm 1 correctly computes the values of A_i by calling Algorithm 2. The only instructions executed are from line 3 to line 6, since v_i has no children. This proves the base step but also, later in the algorithm, the correctness of the values $A_{i,b}$ for any leaf v_i.

Now, given a vertex v_i that is not a leaf (in particular $i > 0$, we assume by induction hypothesis that an optimal solution is computed for each subtree $T_{i_1}, \ldots, T_{i_k}$ rooted at vertices $v_{i_1}, \ldots, v_{i_k}$, children of v_i and for each budget $b = 0, \ldots, B$. These values are stored in A_{i_j}, for each $j = 1, \ldots, k$ (in particular in each row i_j of Table A).

We prove by induction on $j = 0, \ldots, k$ that the values $ST_{j,b,f^+}, b = 0, \ldots, k$ are correct.

For T_i^0, that is the subtree consisting of vertex v_i only, an optimal solution for each possible budget $b = 0, \ldots, B$ is computed from line 3 to line 6, and stored in $ST_{0,b}$. Now, assume that $ST_{j-1,b} = (f^+, f^-, H^+, H^-)$ is correctly computed for the subtree T_i^{j-1}, for each possible b and $j \in \{1, \ldots, k\}$. The discussion in Section 1.3.1 justifies that the line j of Table ST is then properly filled in. By induction, it shows that the line k of Table ST will be properly filled in by Algorithm 2. Since T_i^k is the subtree T_i, it completes the proof of the induction step that Algorithm 1 correctly fills Table A in. This completes the proof of correctness.

Regarding the computational complexity, the execution time of Algorithm 2 is dominated by the instruction at line 9 that requires $O(B)$ time. Since it is repeated $k(B + 1)$ times, Algorithm 2 requires $O(k \cdot B^2)$ time, where k is the number of children for the vertex into consideration. Algorithm 2 is called for each vertex of the tree by Algorithm 1. Then, the overall computational complexity of Algorithm 1 is $O(|E| \cdot B^2)$ and since T is a tree, $O(|E|) = O(|V|)$, and hence the complexity is $O(|V| \cdot B^2)$, which completes the proof. □

We can assume $B \leq n - 2$ since for a larger budget we can cut all edges. So, the complexity of Algorithm 1 is dominated by $O(n^3)$.

For simplicity of the presentation we have described Algorithm 2 with unitary edge costs and vertex values. However, it can easily be generalized to take into account the vertices' weights and the edges' costs as follows.

In the case of generic vertices' weights, Algorithm 2 has to calculate the sum of the weights of the vertices that do not burn, instead of the number of saved vertices, for each subtree. The only required change is in line 6, where $ST_{0,b}$ should be initialized to $(0, \varphi(v_i), \emptyset, \emptyset)$.

With respect to the edges' costs, the Algorithms can be generalized if costs are integer. To this aim, Algorithm 2 should be changed when a cut is needed on the edge e, between v_i and the root of T_{i_j}. If the cost is $c = \kappa(e)$, then the Algorithm should calculate the optimal solutions for a remaining budget $b-c$ (instead of $b-1$), if $b \geq c$. For instance, the condition in line 12 should be changed to $(\pi_i(v_{i_j}) = 0) \wedge (b \geq c)$, and line 13 to $z' \leftarrow \arg\max_{x \in \{0,\ldots,b-c\}} \{ST_{j-1,x,f^+} + A_{i_j,b-c-x,f^-}\}$. Similar changes should be applied to lines 14, 17, 19–22.

Algorithm 1 remains unchanged and the overall complexity $O(|V| \cdot B^2)$ is the same but, this time, we can only assume $B < \sum \kappa(e)$ for integral values of edge costs. So, the overall process is polynomial only for polynomially bounded integral edge costs and pseudo-polynomial otherwise.

These considerations lead to the following theorem.

Theorem 1.3.2. *There exists an algorithm that correctly computes an optimal solution for an instance* $I = (T = (V, E), \mathbb{1}, \pi_i, \kappa, \varphi, B)$ *of* Windy Firebreak Location *where* T *is a tree,* $\kappa(e) \in \mathbb{N}$, $\forall e \in E$ *and* $\pi_i(v) \in \{1, 0\}$. *Its computational complexity is polynomial in* $|V|$ *if* $B = O(Poly(|V|))$ *or* $\kappa(e) = O(Poly(|V|))$.

1.4 The Infinite windy firebreak location problem

In this section, we adapt the model for the Windy Firebreak Location problem, on infinite undirected graphs and introduce the Infinite Windy Firebreak Location problem. The land is modeled as an infinite graph and the goal is to find a cut system that allows the fire to be contained limiting the risk. Given an infinite graph, we assume that a fire ignites in a subset of vertices and propagates to the neighbors. The goal is to select a subset of edges to remove to contain the fire and avoid burning more than a finite part of the graph. Infinite graphs can be seen as a theoretical model of very large lands and then the problem is motivated by preventing a wildfire from escaping, i.e., becoming out of control. In Section 1.4.3 we prove that Infinite Windy Firebreak Location is coNP-complete in restricted cases and we look for cases solvable in polynomial time. We show that Infinite Windy Firebreak Location polynomially reduces to Min Cut for certain classes of graphs like infinite grid graphs and polyomino-grids, a generalization of grids.

1.4.1 Main notations

Here we introduce some notation used in the following sections. Unless otherwise stated, all graphs are infinite and undirected. Note that in an infinite graph, paths are finite and *rays* are the infinite counterpart. So, an infinite graph is connected if every two vertices are linked by a (finite) path.

Let $G = (V, E)$ be an (infinite) undirected graph, for any edge set $H \subset E$, we denote by $G_H = G \setminus H$ the partial graph $(V, E \setminus H)$ obtained from G by removing edges in H. Given a set $V' \subset V$, $G[V']$ denotes the subgraph induced by V' and any graph $G'' = (V'', E'')$, $V'' \subset V, E'' \subset E$ will be called partial subgraph of G. All graph-theoretical terms not defined here can be found in [64].

1.4.2 Problem formalization

The undirected case corresponds to the assumption that all directions of wind are possible. Since the model is meant to be used for fire prevention over a long period of time and not for the response phase, this assumption makes perfect sense. In the finite case, the Windy Firebreak Location problem is defined as selecting a *cut system* $H \subset E$ that minimizes the risk for G_H under a budget constraint. In this section, we will consider that all edge costs are equal and thus, the constraint will be $|H| \leq B$.

In an infinite graph with probabilities of spread all equal to 1 (windy case), we consider only finite cut systems and a finite number of vertices with a positive probability of ignition. Then, all definitions can be easily extended and two cases are to be considered.

First, if all vertices with a positive probability of spread are in finite connected components of G_H, then the risk is finite and immediately computable as the risk associated with the finite graphs consisting in the union of connected components that include at least one vertex of positive probability of ignition. The rest of the graph does not induce any risk. In the second case where there is an infinite connected component of G_H with a vertex of positive probability of ignition, the risk becomes infinite as vertex values have been assumed positive integers. Then, a natural question is whether there is a cut system satisfying a budget constraint and guaranteeing a finite risk. This is the problem we address here. Since this problem does not change with binary probabilities of ignition, we make such assumption. So, the problem is formally defined as follows:

> Infinite Windy Firebreak Location
> Instance: an undirected infinite graph $G = (V, E)$ defined by a finite string of length at most n; a finite subset $\widetilde{V}$, $|\widetilde{V}| \leq n$, of initially burning vertices. A total budget $B \leq n$.
> Question: is there a *cut system* $H \subset E$ such that $|H| \leq B$ and such that the vertices in $\widetilde{V}$ are in finite connected components of G_H (the fire can be contained)?
>
> We will denote such an instance $(G, \widetilde{V}, B)$ and call n the *size* of G.

1.4.3 The complexity of Infinite Windy Firebreak Location

To our knowledge, there have been very few attempts to extend the definition of complexity for the case of combinatorial problems defined on infinite graphs. Among these attempts, [27] considers instances that are defined with incomplete information. Here, we adopt a completely different perspective by considering finitely represented infinite graphs. This means that we assume a finite encoding of each instance. Then, through a given encoding scheme, the problem becomes a finite combinatorial problem in common sense. The size of an instance is then the length of the finite string representing it or any polynomial function of this length. This gives us the possibility to refer to the classical complexity theory to analyze the intractability of problems on finitely represented infinite graphs. In this process, however, we need to be careful that different encoding schemes lead to different problems with, possibly, different complexity [76], as the example in the next section will show.

Here, we give some evidence of the hardness of Infinite Windy Firebreak Location, even on a very simple class of finitely represented infinite graphs. The graphs we consider are constituted by a finite star with non-crossing infinite rays (called *infinite tail*) attached to some leaves of the star. For such a graph, we denote o as the center of the star. Only the center o has a probability of ignition equal to 1 and all other vertices have a probability of ignition equal to 0.

A trivial finite representation is by listing the neighbors of o and indicating those that have an infinite tail. So, a natural representation is a boolean vector of dimension n, where n is the number of neighbors of the center o and 1 entries correspond to infinite tails. With this representation, a reasonable size of such an instance is the degree of o. Within this encoding scheme, the problem is trivially polynomially solvable: the size of a minimum cut is the number of neighbors of o with an infinite tail. We can also represent such an instance as two numbers, the number of neighbors of o with an infinite tail and the number of neighbors of o without an infinite tail. The related size is then the number of bits required to represent these numbers; it is a logarithm of the previous size and the problem remains clearly polynomial within this representation.

We now propose a subclass of these instances with an alternative representation. Assume that we have a finite set X of size n and a boolean function $f : 2^X \to \{0, 1\}$ computable in polynomial time with respect to n, where 2^X is the set of subsets of X. The neighborhood of o is 2^X and only those neighbors x such that $f(x) = 1$ have an infinite tail. Since we can decide in polynomial time whether a neighbor of o has an infinite tail, it is reasonable to define n as the size of the graph. The center o is still the only vertex on fire at the start ($\widetilde{V} = \{o\}$) and B polynomially bounded in the size n. We denote by $\mathcal{S}$ the set of these instances with this representation.

Proposition 1.4.1. Infinite Windy Firebreak Location *restricted to instances in $\mathcal{S}$ is coNP-complete.*

Proof. Note first that this particular case of Infinite Windy Firebreak Location is in coNP. Consider indeed an instance $(G, \widetilde{V} = \{o\}, B)$ in $\mathcal{S}$: G is a star with center o defined from a set X of size n and a boolean function f. It is a no-instance if and only if we have $B + 1$ different neighbors of o with an infinite tail. Given $B + 1$ neighbors of o, $x_0, \ldots, x_B$ we can check in polynomial time whether they are all different and whether $\forall i \in \{0, \ldots, B\}, f(x_i) = 1$.

We consider an instance I of SAT, known to be NP-complete, with a set X of n variables and m clauses. Without loss of generality, we can assume $m \leq n$: we indeed just can add to X m artificial variables and one clause including all of them. We associate to it the graph G obtained by linking the center o with all truth assignments (in one-to-one correspondence with 2^X). For any truth assignment x, $f(x) = 1$ if and only if all clauses are satisfied; f is computable in polynomial time. We also add to o an infinite ray that does not cross any tail. We then consider the instance $I' = (G, \widetilde{V} = \{o\}, B = 1)$ of Infinite Windy Firebreak Location. I' can be defined in polynomial time with respect to n and is an instance in $\mathcal{S}$. It is a no-instance if and only if I is a yes-instance. This concludes the proof. □

Note that, in the class of instances $\mathcal{S}$, only one vertex - the center - has a non-zero probability of ignition. If we do not require this property, then exactly the same

proof can be applied on graphs consisting of $2^{|X|}$ disjoints components, each being either a single vertex or a ray.

1.4.4 Some cases solvable in polynomial time

The hardness results in the previous section motivate the question of identifying some cases solvable in polynomial time for Firebreak Location. Since Infinite Windy Firebreak Location and Windy Firebreak Location revealed to be hard in restricted cases and since the complexity of Windy Firebreak Location with binary ignition probabilities is still open, it seemed to us relevant to start with this case. We identified two cases solvable in polynomial time and possibly the methods could be extended to other cases. For some graph classes including grids, the infinite version of Windy Firebreak Location turns to be polynomial since it reduces to Min Cut. Roughly speaking, it means that deciding whether we can contain the fire (i.e., deciding whether at least a finite risk can be guaranteed) instead of minimizing the risk is polynomial. This case is also interesting since it is not impacted by restrictions on the vertex values, edge costs and ignition probabilities. So, it is enough to consider the case where all these parameters are binary.

Infinite Windy Firebreak Location in Infinite Grids

In this subsection, we identify a class of Infinite Windy Firebreak Location instances that are polynomially solvable. Complexity considerations for Infinite Windy Firebreak Location will refer to n assumed to be at least $|\widetilde{V}| + B$, as the size of the instance, where $\widetilde{V}$ is the set of vertices with a positive probability of ignition. Note that the problem is not changed if we assume all probabilities of ignition equal to 1 in $\widetilde{V}$ (and 0 elsewhere).

We outline two properties of infinite graphs that are in particular satisfied by various versions of infinite grids. In an infinite connected graph $G = (V, E)$ and any subgraph $G[V']$ of G, we call *escaping edges* from $G[V']$ any edge between V' and an infinite connected component of $G[V \setminus V']$. We call *ball centered on vertex x and of radius $K \in \mathbb{N}$* in G the set of vertices $\{y \in V, d(x, y) \leq K\}$.

> Polynomial growth property:
> The first property, called *polynomial growth property* states that the cardinality of balls for the minimum path distance (all edge lengths are 1) is polynomial with respect to the radius. It expresses that the graph has a "polynomial expansion" around any vertex. This property was first introduced in [128].

Expansion property:
On the contrary, the second property, called *expansion property*, expresses that the graph always expands around vertices: there is an integral polynomial function L such that, for any value B, any finite subgraph with more than $L(B)$ vertices has at least $B+1$ escaping edges.

We are interested in graphs satisfying both properties. Then, the same polynomial function can be used to describe the properties, as outlined in the following remark:

Remark 1.4.2. *If an infinite graph G satisfies the polynomial growth and the expansion properties, then there is a polynomial function L such that:*

(i) $\forall x \in V, \forall K \in \mathbb{N}, |\{z, d(x,z) \leq K\}| \leq L(K)$;

(ii) Any finite connected subgraph of size more than $L(B)$ has at least $B+1$ escaping edges.

Proof. Indeed, both properties are still valid if we replace the polynomial function with a larger one. We conclude by noticing that the maximum between two polynomial functions is a polynomial function. □

About graphs satisfying the polynomial growth and expansion properties

As outlined by the following lemmas, these two properties are satisfied in many classes of infinite graphs that are natural in our application context.

Infinite grids correspond to the simplest illustration. Let a *double ray* be the graph $P = (\mathbb{Z}, E)$ with $E = \{\{i, i+1\} : i \in \mathbb{Z}\}$. The infinite grid is then defined as the Cartesian product $P \times P$. It is a non-directed graph.

Lemma 1.4.3. *The infinite grid satisfies the polynomial growth property and the expansion property.*

Proof. It satisfies the polynomial growth property: for any vertex x of the infinite grid and any integer K, we have: $|\{z, d(x,z) = K\}| = 4K$ and consequently, each ball of radius K has cardinality $1 + 2K(K+1)$.

It is also easy to verify that the infinite grid satisfies the expansion property. In [85] it is proved that the minimum possible perimeter of a polyomino with p tiles is $2\lceil 2\sqrt{p} \rceil$. The adjacency graph (or *dual graph*) of a polyomino, where tiles are associated with vertices and tiles adjacency corresponds to vertex adjacency, is a finite subgraph of the infinite grid. Conversely, every finite subgraph of the grid is the adjacency graph of a polyomino. Several polyominoes may have isomorphic adjacency graphs. However, we can choose the embedding of the adjacency graph in the grid that preserves the orientation: two adjacent tiles one of the right of

(respectively above) the other correspond to two vertices in the grid with the same relative position. Then, the correspondence is one-to-one up to a translation and the external perimeter of the polyomino corresponds to the number of escaping edges of the corresponding subgraph of the infinite grid. So, the result of [85] is equivalent to say that a finite subgraph of the infinite grid with p vertices has at least $2\left\lceil 2\sqrt{p}\right\rceil$ escaping edges. Choosing $p = \frac{(B+1)^2}{16}$ ensures at least $B+1$ escaping edges. So, in the infinite grid we can choose for instance $L(B) = \left\lceil \frac{B(B+2)}{16} \right\rceil$. This concludes the proof. □

It is straightforward to verify that, if an infinite graph satisfies the polynomial growth property, then any partial subgraph also does. Indeed, balls of the partial subgraph are always contained in balls of the original graph.

More work is required to analyze the expansion property in a subgraph. When considering infinite subgraphs of an infinite graph represented by a finite string, we will only consider removing a finite number of vertices to ensure that the new graph can also be represented by a finite string. Then, it will be natural to consider that the description of the removed vertices is part of the description of the subgraph and consequently, the size of the subgraph is at least the number of removed vertices. This leads to the surprising fact that the size does not decrease but may increase when taking a subgraph. Since Infinite Windy Firebreak Location is defined in infinite graphs, we will not consider finite subgraphs of an instance as a new instance. With these definitions, the expansion property is also transferred to subgraphs.

Lemma 1.4.4. *If an infinite graph of finite maximum degree Δ satisfies the expansion property for a polynomial function L, then any induced subgraph obtained by removing a finite set V' of vertices also satisfies the expansion property for the polynomial function $L' : B \mapsto L(B + \Delta|V'|)$.*

Proof. Consider an infinite graph $G = (V, E)$ of finite maximum degree Δ satisfying the polynomial expansion property for the polynomial function L and let V' be a finite subset of V. We prove that $G[V \setminus V']$ also satisfies the polynomial expansion property. Consider a finite subgraph $G'' = G[(V \setminus V') \cap V'']$ of $G[V \setminus V']$ with not more than B escaping edges in $G[V \setminus V']$. Then, G'' has at most $B + \Delta|V'|$ escaping edges in G since each vertex of V' cannot induce more than Δ new escaping edges. As a consequence, G'' is of order at most $L(B+\Delta|V'|)$. Since $|V'|$ and Δ are constant for a fixed subgraph, L' is a polynomial function for the variable B. This completes the proof. □

Finally, we outline that, adding edges between vertices at bounded distance also preserves both properties. Adding edges to an infinite graph corresponds to the union of two infinite graphs on the same vertices. If both graphs are represented by

finite strings, then so does the union and the size of the union can be set as the sum of sizes of the two infinite graphs.

Lemma 1.4.5. *Let G be an infinite graph of size n that satisfies the polynomial growth and the expansion properties. Let G' be obtained from G by adding edges between vertices at distance at most S for a constant S. Then, G' satisfies the polynomial growth and the expansion properties.*

Proof. Using Remark 1.4.2, we suppose that G satisfies both properties for the same polynomial function L.

Two vertices at distance K in G' are at distance at most $S \times K$ in G. So, a ball of radius K in G' is of cardinality at most $L(S \times K)$, which is a polynomial in K.

For the expansion property, we consider, for some B, a finite set of vertices, V', with $|V'| > L(B)$. In G, there are more than B escaping edges and thus, this is true as well in G', which completes the proof. □

Using Lemmas 1.4.3 and 1.4.5, we deduce in particular that infinite grids with all diagonals $((x, y), (x + 1, y + 1)), ((x, y), (x + 1, y - 1))$ or a finite number of them satisfy both properties and can be represented by a finite string.

We conclude this section with a generalization of infinite grids that satisfy both properties. Consider any tiling of the two dimensional plan with polyominoes of size at most S unit-squares, for a fixed constant S and that can be represented by a finite string. Then, we call *Polyomino-grid* the adjacency graph of the different polyominoes in such a tiling. It is an infinite graph represented by a finite string and the length of this string is the size of this graph. Usual grids correspond to the case $S = 1$. A wall is a case where $S = 2$.

Proposition 1.4.6. *Polyomino-grids satisfy the polynomial growth property and the expansion property.*

Proof. Given a polyomino-grid G and the related tiling of the plan, partitioning each polyomino associated with a vertex in at most S unit-squares leads to the regular tiling with squares. Given two vertices x and y, and two squares s_x and s_y in the polyomino associated with x and y, respectively. Then, in the infinite grid, the vertices associated with s_x and s_y are at distance at most $S \times K$. As a consequence, the cardinality of a ball of radius K in G is at most the cardinality of a ball of radius $S \times K$ in the infinite grid. As a consequence, using Lemma 1.4.3, G satisfies the polynomial growth property.

Suppose now a finite connected subgraph G' of G with p vertices. Partitioning as previously each polyomino into at most S unit-square leads to a connected polyomino with at least p and at most $p \times S$ squares, thus a connected subgraph G'' with at least p and at most $p \times S$ vertices in the infinite grid. Using Lemma 1.4.3, there is a polynomial function L such that, if, for $B \in \mathbb{N}$, $p > S \times B$, then the number of

escaping edges in G'' is greater than $S \times B$. Each escaping edge in G' corresponds to at most S escaping edges in G'' and consequently, the number of escaping edges in G'' is greater than B, which concludes the proof. □

Lemma 1.4.4 ensures that removing a finite number of vertices from a polyomino-grid does not affect the two properties. Lemma 1.4.5 ensures we can add edges between vertices at a bounded distance. The resulting classes of graphs are relevant as a fire spread network in wildfire emergency context. Polyomino-grids appear naturally as adjacency graphs of areas of similar surface in a landscape, removing some vertices allows to represent zones where the fire will not spread (like lakes) and adding edges between vertices that are close allows to represent spread by ember in some areas.

In the next section, we outline that Infinite Windy Firebreak Location can be solved in polynomial time in polyomino-grids.

A case solvable in polynomial time for Infinite Windy Firebreak Location

We then denote $\mathcal{G}_G$ the class of Windy Firebreak Location instances of the form $I = (G[V \setminus V'], B)$, where G is a finitely represented connected infinite graph of finite degree Δ and where Δ, $|V'|$, $|\widetilde{V}|$ and B are bounded by the size of I.

Note that $G[V \setminus V']$ may have finite connected components. However, we do not change the nature of I by adding to V' all vertices of a finite connected component of $G[V \setminus V']$. We just need to remark that the sum of cardinalities of these finite connected components is polynomial and that these components can be computed in polynomial time with respect to n:

Lemma 1.4.7. *Denote C the set of vertices of all the finite connected components of $G[V \setminus V']$; C is finite of cardinality at most $L(\Delta \times |V'|)$ and can be listed in polynomial time with respect to n.*

Proof. Since G is connected, any escaping edge from $G[C]$ in G is adjacent to V' and consequently, their number is at most $\Delta \times |V'|$. This implies, using the expansion property, that $|C| \leq L(\Delta \times |V'|)$. Since all connected components of $G[C]$ are adjacent to V' and the maximum degree is Δ (a constant), C can be listed using Breadth First Search from each vertex $x \in V'$. If the search reveals a connected component of at least $L(\Delta \times |V'|) + 1$ vertices, then it is an infinite connected component and the search from x is stopped. In all, the complexity is $O(|V'| \times \Delta \times L(\Delta \times |V'|))$. □

So, given Lemma 1.4.7, we can assume that $G[V \setminus V']$ has only infinite connected components. This requires increasing the size of the new instance to $\max(n, |V'| + |C|)$ but this does not affect whether algorithms are polynomial or not.

Theorem 1.4.8. *Consider a connected infinite graph G of finite maximum degree that satisfies the expansion property and the polynomial growth property. Then,* Infinite Windy Firebreak Location *is polynomial on the class $\mathcal{G}_G$.*

Proof. Using Remark 1.4.2, we assume that the same polynomial function L is used in the polynomial growth property and the expansion property. We denote Δ as the maximum degree of G. We reduce Infinite Windy Firebreak Location on $\mathcal{G}_G$ to the problem of finding a minimum capacity (s,t)-cut, denoted Min Cut, in a transportation network N of polynomial size w.r.t. n, the size of G. Since Min Cut is polynomially solvable [74], it will complete the proof.

Consider $I = (G[V \setminus V'], \widetilde{V}, B)$, a Infinite Windy Firebreak Location instance of size n, where $|V'| \leq n, |\widetilde{V}| \leq n$, and $B \leq n$.

As seen before, we assume that $G[V \setminus V']$ has only infinite connected components. We then consider the set $V'' = \{x \in V, d(x, \widetilde{V}) \leq L(B + \Delta|V'|)\}$, where d denotes the distance in G. We then consider the infinite graph $G[V \setminus (V' \cup V'')]$ and denote V''' the set of vertices of all finite connected components of $G[V \setminus (V' \cup V'')]$.

We define the transportation network N by adding to $G[(V'' \cup V''') \setminus V']$ a source s and all edges $sx, x \in \widetilde{V}$. Similarly, we add a vertex t and all edges from any vertex incident to an escaping edge from $G[(V'' \cup V''') \setminus V']$ in $G[V \setminus V']$ to t. All edges in N incident to s or t have capacity $B + 1$. All edges of $G[(V'' \cup V''') \setminus V']$ have capacity 1. With this capacity system, a (s,t)-cut of capacity at most B cannot include any edge incident to s or t.

By definition, $V'' = \cup_{x \in \widetilde{V}} \{z, d(x,z) \leq L(B + \Delta|V'|)\}$ and consequently, using the polynomial growth property of L, $|V''| \leq |\widetilde{V}| \times L\left(L(B + \Delta|V'|)\right)$, which is polynomially bounded w.r.t. n. In addition, V'' can be listed in polynomial time using Breadth First Search from each vertex in $\widetilde{V}$. Lemma 1.4.7 (replacing V' with $V' \cup V''$) guarantees that $|V'''| \leq L\left(\Delta \times (|V'| + |V''|)\right)$, which is polynomial, and V''' can be listed in polynomial time. We deduce that N is of polynomial order at most

$$|\widetilde{V}| \times L\left(L(B + \Delta|V'|)\right) + L\left(\Delta \times \left(|V'| + |\widetilde{V}| \times L(L(B + \Delta|V'|))\right)\right) + 2,$$

which is polynomial w.r.t. n as a composition of polynomial functions. In addition, N can be computed in polynomial time since G is represented in polynomial time and $V'' \cup V'''$ and V' can be listed in polynomial time.

We then claim that:

> There is, in N, a (s,t)-cut of capacity at most B if and only if I is positive,

which will conclude the proof.

Assume first there is a (s,t)-cut of capacity at most B and denote (X_s, X_t) the two parts:$\{s\} \cup \widetilde{V} \subset X_s$; similarly, t and all vertices incident to t in N are in X_t. The number of edges between X_s and X_t in G is at most B. It corresponds to a cut system H. Any path in $G[(V'' \cup V''') \setminus V']$ from $\widetilde{V}$ to X_t includes at least one edge from H. Consider a vertex $x \in \widetilde{V}$ and the related connected component C_x

in $G[V \setminus V'] \setminus H$. Consider, in $G[V \setminus V']$, a ray starting from x. Since $V'' \cup V'''$ is finite, this ray gets out $V'' \cup V'''$ and let z^+ be the first vertex from x along this ray such that $z^+ \notin (V'' \cup V''')$. Let z^- be the vertex just before z^+. The corresponding path from x to z^- is in $(V'' \cup V''') \setminus V'$ and, by definition of V''', the edge z^-z^+ is escaping from $G[(V'' \cup V''') \setminus V']$ in $G[V \setminus V']$. So, $z^- \in X_t$ and consequently the path from x to z^- includes at least one edge from H. This means that any ray from x in $G[V \setminus V']$ crosses an edge from H. This holds for any $x \in \widetilde{V}$; H is a cut system that allows containing the fire and I is positive.

Assume conversely that I is positive and let H be a cut system with at most B edges that allows to contain the fire. Consider as previously a vertex $x \in \widetilde{V}$ and the related connected component C_x in $G[V \setminus V'] \setminus H$. C_x has at most B escaping edges and consequently, using Lemma 1.4.4, we have $|C_x| \leq L(B + \Delta|V'|)$. In particular, the diameter of C_x is at most $L(B+\Delta|V'|)-1$. Consequently, edges in H are edges of $G[V'']$ and moreover, all paths from $\widetilde{V}$ to t in N cross at least one edge of H. It means that H is a (s,t)-cut in N, which concludes the proof. □

Using Proposition 1.4.6, we deduce:

Corollary 1.4.9. Infinite Windy Firebreak Location *can be solved in polynomial time in polyomino-grids.*

From an instance $I = (G[V \setminus V'], \widetilde{V}, B)$ of Infinite Windy Firebreak Location, we build the network N and use a minimum cut algorithm to solve Infinite Windy Firebreak Location, using Theorem 1.4.8. The minimum cut algorithm runs in $O(nm^2)$ [74] in a graph with n vertices and m edges. Then the complexity is of order $O\left((|V''| + |V'''|)^3\Delta^2\right) \subset O\left((|V''| + |V'''|)^5\right)$, where $|V''| = |\widetilde{V}| \times L\left(L(B + \Delta|V'|)\right)$, $|V'''| = L\left(\Delta \times \left(|V'| + |\widetilde{V}| \times L(L(B + \Delta|V'|))\right)\right)$ and Δ is the maximum degree of the graph $G[V \setminus V']$.

1.5 Concluding remarks

In this chapter, we studied the computational complexity of Firebreak Location and its restricted version Windy Firebreak Location. Both problems are motivated by a wildfire management context. We focused on specific instances, in particular low-degree planar graphs, relevant for practical application. We proved that Windy Firebreak Location is still NP-complete in bipartite planar graphs of maximum degree 4 and unitary vertex values and edge costs. On the other hand, we proved that Windy Firebreak Location is polynomial on trees with polynomially bounded edge costs and binary probabilities of ignition.

The hardness results motivate studying the approximation properties of the problem and identifying new classes of instances solvable in polynomial time. Altogether

these results would help better understand the problem. We introduced the Infinite Windy Firebreak Location problem. The land is modeled as an infinite graph, and the goal is to find a cut system that allows the fire to be contained, limiting the risk. Infinite graphs can be seen as a theoretical model of very large lands. The problem is motivated by preventing a wildfire from escaping, i.e., becoming out of control. We showed that the problem is coNP-complete in restricted cases. This motivates the search for cases solvable in polynomial time. We outlined two properties of infinite graphs: the polynomial growth property and the expansion property. These are satisfied by various versions of infinite grids, as well as a generalization called Polyomino-grids. Polyomino-grids naturally represent a land with areas of similar surfaces and also allow representing fire spread by embers by adding edges between close areas. We showed that in these cases Infinite Windy Firebreak Location is polynomial and reduces to the problem of finding a Min Cut in a transportation network for graphs satisfying both the polynomial growth property and the expansion property.

Chapter 2

Model validation and heuristics

Given the computational complexity of Firebreak Location, heuristics methods are the most appropriate to handle it for practical applications. The hardness results motivate studying the approximation properties of the problem and identifying new classes of instances solvable in polynomial time. In this chapter, we describe a heuristic algorithm that computes the risk for each vertex and the total risk of the graph. This algorithm is implemented in the web application presented in Section 2.4, while in Section 2.2 we present a particular case of the Firebreak Location problem in which all the probabilities of ignition can be considered equivalent. In Section 2.3, we present the model validation. We instantiate the graph model on the territory of Cap Corse, the peninsula of Corsica island to prove the usability of the model. We partition the peninsula of Cap Corse into adjacent areas connected by edges. Then, we estimate the probabilities of ignition using data on historical fires and we estimate the probabilities of spread by running several fire simulations. We show data directly on geographical maps for an intuitive understanding of the areas with a higher risk of fire. Finally, in Section 2.4, we present a prototype web application. The goal is to give a tool to target end-users, i.e. fire and risk managers, to effectively visualize data about wildfire and to simulate interactively risk mitigation interventions, quantifying their effect, in terms of risk reduction, before deployment.

2.1 Approximated risk calculation

Risk computation, in the general case, is #P-hard, and its approximability is still an open problem. In this subsection, we then describe a heuristic algorithm to compute the risk, adapted from [136, 137].

The algorithm first adds a universal vertex f that represents the fire. Then, it connects f to all nodes of the original graph. The probability of spread for each of these edges is then defined equal to the probability of ignition of the connected node, i.e., given node e from f to v, $\pi_s(e) = \pi_i(v)$.

Given this new graph, the algorithm uses discrete time and starts at time t_0. At the time t_0, only node f is burning. Then, for each edge e between f and node v, the algorithm compares the probability of spread $\pi_s(e)$ to a uniformly distributed random number $r \in [0, 1)$, so to simulate a possible ignition in v. Accordingly, we consider that a fire starts in v, and then v is burnt, if $r < \pi_s(e)$. Then, $t = t + 1$. At the time t, the algorithm takes into account all nodes v' that were reached by fire at time $t - 1$. Then, for each edge e between v' and v, where v is a non-burnt node, similarly as above, the algorithm compares the probability of spread $\pi_s(e)$ to a uniformly distributed random number $r \in [0, 1)$, so to simulate the possible spread from v' to v. The algorithm ends when no new nodes are burnt.

Such a process is iterated until the standard deviation of the number of burnt nodes becomes lower than a certain threshold. The chosen threshold ϵ is such that the following inequality is satisfied:

$$2 \cdot x_{\alpha/2} \cdot \sqrt{\frac{S_N^2(t)}{N}} < \epsilon \tag{2.1}$$

where N is the current iteration number, $S_N^2(t)$ is the estimate variance of the number of burnt nodes, $1 - \alpha$ is the confidence level, and $x_{\alpha/2}$ is chosen so that $\int_{-\infty}^{x_{\alpha/2}} g(t)dt = 1 - \alpha/2$, where $g(t)$ is the Gaussian standard density. Basically, the inequality aims at calculating the confidence intervals for the number of burnt nodes.

At the end of all the required iterations:

- the probability of burning for each node v is calculated as the number of times that v burned, divided by the number of iterations;
- the risk for each node v is calculated as the probability of burning for v, multiplied by the node value $\omega(v)$;
- the risk is calculated as the sum of all risks associated to each node v.

2.2 Partitioning problem

In this section, we consider the particular instance of Firebreak Location in which all the probabilities of ignition are equal to a fixed value p, all the vertices have the same value, all ignition events are independent, the graph is symmetric (if $xy \in E$, then $yx \in E$) and probabilities of spread are all set to one. This setting can model a large uniform territory divided in subareas of approximately the same dimension and with no specific predominant wind direction during the year. Moreover, we investigate the propagation phenomenon in the worst case that is when the fire ignited in a vertex of the graph certainly propagates to the adjacent nodes (probabilities of spread all equal to 1). As in the general problem, the spreading of fire can be mitigated with the construction of firebreaks modeled by the suppression of connections in the graph. The goal is always to identify a subset H of edges to be cut under a budget constraint. Under this setting, if a fire starts on a node (i.e., in an area), then only the corresponding connected component in $G \setminus H$ will burn.

2.2.1 Minimizing the worst case scenario for a single fire

We are first interested in the problem of minimizing the maximum number of nodes that may burn if only one node ignites. This corresponds to mitigating the worst-case scenario with a single fire. From a graph perspective, the objective is to minimize the maximum number of nodes in a connected component after suppression of the cut H. Suppose we fix the number k of connected components of $G \setminus H$, then, without the budget constraint, the maximum size of the k connected components is minimized if all connected components have the size $\left\lfloor \frac{|V|}{k} \right\rfloor$ or $\left\lceil \frac{|V|}{k} \right\rceil$. Then, the problem reduces to the following graph partition problem:

> k-Graph Partition
> Instance: A non-directed graph $G = (V, E)$; for every edge $e \in E$ a cost $\kappa(e)$; an integer $k \leq |V|$.
> Solution: A cut system $H \subset E$ such that $G \setminus H$ has k connected components all of size $\left\lfloor \frac{|V|}{k} \right\rfloor$ or $\left\lceil \frac{|V|}{k} \right\rceil$.
> Objective: Minimize $\kappa(H)$.

Note that in a non-directed graph, a cut system is just any set of non-directed edges.

2.2.2 Minimizing the total risk

Under our settings, the risk $\rho(G_H)$ of a cut system $H \subset E$ is proportional to the expected number of burned nodes. In this part, we give evidence that, for a small

probability of ignition and without budget constraint, minimizing the total risk for a graph fragmented into k components leads to the same solution as above. More precisely, we consider a cut system H that fragments the graph into $k \geq 2$ connected components of cardinality $x_1, \ldots, x_k$ and then, we justify that a balanced solution with all connected components of the same size is near optimal.

In the case where all x_is are integers and under the assumptions we made, the expected number of burned nodes is

$$R(x_1, \ldots, x_k) = \sum_{i=1}^{k} x_i(1 - (1-p)^{x_i}) \tag{2.2}$$

and the variables x_is satisfy the constraint

$$\sum_{i=1}^{k} x_i = |V|. \tag{2.3}$$

We use a continuous optimization argument, relaxing the constraints that x_is are integers into $x_i > 0, i = 1, \ldots, k$. In addition, if x_is are positive integers and satisfy Equation 2.3, then we have $x_i \leq |V| - k + 1 < |V| - k + 2, i = 1, \ldots, k$. We then show that, for small p, the solution $(x_i = \frac{|V|}{k})_{i=1,\ldots,k}$ minimizes $R(x_1, \ldots, x_k)$ among all $(x_1, \ldots, x_k)$ satisfying the constraint 2.3. Even though it does not give formal proof for the discrete case, it supports strong evidence that a balanced fragmentation is optimal. In particular, in a relatively large graph, this approximation is perfectly justified.

To simplify expressions, we denote $\bar{p} = 1 - p$, the unique probability of non-ignition in a single area. Note that, using this relation, we have $R(x_1, \ldots, x_k) = |V| - \sum_{i=1}^{k} x_i \bar{p}^{x_i}$ and consequently, our relaxed optimization problem is equivalent to:

$$(P_k) : \begin{cases} \max & \bar{R}(x_1, \ldots, x_k) & = \sum_{i=1}^{k} x_i \bar{p}^{x_i} & \\ s.t. & \sum_{i=1}^{k} x_i & = |V| & \\ & x_i & < |V| - k + 2 & i = 1, \ldots, k \\ & x_i & > 0 & i = 1, \ldots, k. \end{cases}$$

Proposition 2.2.1. *For any integer $2 \leq k \leq |V|$, if $p \leq 1 - e^{-\frac{2}{|V|-k+2}}$, then $(x_i = \frac{|V|}{k})_{i=1,\ldots,k}$ is an optimal solution of (P_k).*

Proof. Assume $p \leq 1 - e^{-\frac{2}{|V|-k+2}} < 1$, so $\bar{p} > 0$.
We denote $f(x) = x\bar{p}^x = xe^{x \ln \bar{p}}$. The function f is infinitely derivable. Its first derivative is $f^{(1)}(x) = \bar{p}^x(1 + x \ln \bar{p})$ and its second derivative is $f^{(2)}(x) = (\ln \bar{p})\bar{p}^x(2 + x \ln \bar{p})$.

We then consider the Lagrangian function

$$L(x_1, \ldots, x_k, \lambda) = \bar{R}(x_1, \ldots, x_k) + \lambda \left(\sum_{i=1}^{k} x_i - |V| \right).$$

Denote $x^* = \frac{|V|}{k}$ and note that

$$\left[x_i = x^*, i = 1, \ldots, k, \lambda = -\bar{p}^{x^*}(1 + x^* \ln \bar{p}) \right]$$

is a critical point for the Lagrangian L (all partial derivatives are 0). In addition note that, since $p \leq 1 - e^{-\frac{2}{|V|-k+2}}$, $f^{(2)}(x) < 0$ for any x in the domain of (P_k) (note that $\ln(\bar{p}) < 0$) and consequently, $(x_i, \ldots, x_k) \mapsto L(x_1, \ldots, x_k, \lambda)$ is concave as sum of concave functions. Thus, the considered critical point is the only optimal solution of (P_k) [100]. □

Note that the threshold slightly increases with k and consequently, if $p \leq 1 - e^{-\frac{2}{|V|}}$, it satisfies the condition for any $k \geq 2$. Note as well, it is near linear in $\frac{1}{|V|}$ for large values of $|V|$ since, in this case, $1 - e^{-\frac{2}{|V|}} \sim \frac{2}{|V|}$ but this approximation is accurate even for moderate values of $|V|$, as outlined in the Table 2.1. In many cases, it will be relevant to limit a priori the size of each part and in this case, the threshold is higher. For instance, if we impose $x_i < \frac{|V|}{2}, i = 1, \ldots, k$, the threshold becomes $1 - e^{-\frac{4}{|V|}}$. Table 2.1 gives values of the threshold for different values of p between 10 and 200.

$\|V\|$	$1 - e^{-\frac{2}{\|V\|}}$	$1 - e^{-\frac{2}{\|V\|-k+2}}$ k=3	$1 - e^{-\frac{2}{\|V\|-k+2}}$ k=5	$1 - e^{-\frac{4}{\|V\|}}$	$\frac{2}{\|V\|}$
10	18.13%	19.93%	24.85%	32.97%	20.00%
20	9.52%	9.99%	11.10%	18.13%	10.00%
30	6.45%	6.66%	7.14%	12.48%	6.67%
40	4.88%	5.00%	5.26%	9.52%	5.00%
50	3.92%	4.00%	4.17%	7.69%	4.00%
60	3.28%	3.33%	3.45%	6.45%	3.33%
70	2.82%	2.86%	2.94%	5.55%	2.86%
80	2.47%	2.50%	2.56%	4.88%	2.50%
90	2.20%	2.22%	2.27%	4.35%	2.22%
100	1.98%	2.00%	2.04%	3.92%	2.00%
150	1.32%	1.33%	1.35%	2.63%	1.33%
200	1.00%	1.00%	1.01%	1.98%	1.00%

Figure 2.1: Some values of the threshold with different assumptions

2.2.3 How to use k-Graph Partition

Note that, under the conditions of Proposition 2.2.1, the minimum expected number of burned nodes for a graph fragmented into k components is $R(x^*, \ldots, x^*) = |V|(1-\bar{p}^{\frac{|V|}{k}})$ and that this value decreases if k increases. Similarly, the maximum number $\left\lceil \frac{|V|}{k} \right\rceil$ of burned nodes decreases in k. Whichever criteria we choose, a possible approach for our original problem with a budget B is to compute, or approximate, the optimal value $\kappa^*(k)$ of k-Graph Partition for incremented values of $k \geq 2$ and then choose $k^* = \max\{k, \kappa^*(k) \leq B\}$. If, for the second criteria (expected number of burned vertices) the threshold for p (see Table 2.1) still seems reasonable, then we select the obtained solution as an approximation of original problem.

Even though the conditions to apply this strategy are quite restrictive, it provides a general solution independent of the probability of ignition p as far as it is small enough. When the probability of ignition is hard to evaluate, in particular on lands with a low fire history but now at risk due to climate change or recent changes in the environment, such a simplified model independent of the probability can be useful compared to a more sophisticated model based on poor quality data. In addition, the fact that the proposed solution jointly optimizes both criteria lends additional credence to it and motivates addressing the k-Graph Partition problem. In what follows, we investigate a possible heuristic to solve k-Graph Partition and use it on a study case in Corsica (France).

2.2.4 Strategy to solve k-Graph Partition

Given a graph $G = (V, E)$ with weight $\kappa(e)$ for each edge $e \in E$ and positive integers $U \leq |V|$ and Y, determining whether there is a partition of V in two disjoint sets V_1, V_2, such that $|V_1| \leq U$, $|V_2| \leq U$ and the sum of the weights of the edges between V_1 and V_2 is no more than Y is an NP-complete problem. In addition, this hardness result still holds if $U = |V|/2$ (i.e., we want to obtain two perfectly balanced subsets of nodes) and $\kappa(e) = 1$ for all $e \in E$ [77]. As a consequence, the restriction of 2-Graph Partition where $\forall e \in E, \kappa(e) = 1$ is NP-hard. To our knowledge, the hardness of k-Graph Partition in planar graphs is still open but there is strong evidence that it might be hard for this class [68].

Given the hardness of the problem, partitioning the graph in balanced components can be addressed using heuristics. One of the most efficient techniques is multi-level partitioning [92]. It is carried out in three phases. The first phase, called coarsening, iteratively reduces the size of the graph by grouping together nodes that are close. This algorithm generates a sequence of graphs starting with the original graph and where each graph in the sequence is a coarsened versions of the previous graph. The second phase, called partitioning, starts when the coarsened graph is small enough and can be easily split into k parts using an exhaustive search or other exact

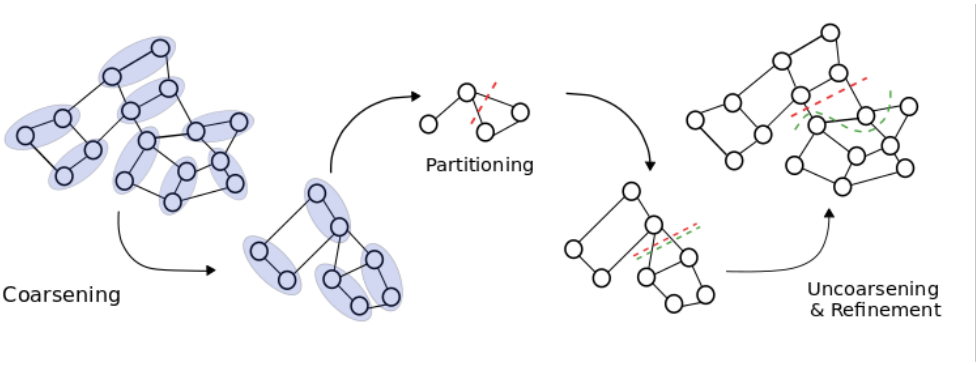


Figure 2.2: Summary of multi-level partitioning algorithms on a weighted graph. From the left, the figure shows the coarsening phase in which close nodes are grouped together; in the center, the smaller graph is partitioned; on the right is shown the third phase of the algorithm in which the graph step by step grows back to the original size and the value of the cut is improved. In the last step, the green cut, representing the refinement phase, is an improvement of the red cut.

algorithm. In the third phase, called uncoarsening and refinement, the obtained partition is iteratively projected backwards on the previous graph in the sequence and improved, at each step, using a local search.

2.2.5 Multi-level partitioning simulations and results

We have tested this technique on the geographical area of the North of Corsica (see Figure 2.3). For this example, we have used the METIS library that proposes an implementation of a multi-level partitioning algorithm [92]. The land is represented with a graph of 236 nodes and has been partitioned into an increasing number of balanced components. The plot was obtained with a probability of ignition $\pi_i = 0.0084$ that is the threshold probability for $k = 2$ parts. The results are shown in Figure 2.3: the red plot shows the risk and the black plot shows the related cost for partitions with an increasing number k of parts (horizontal axis).

The overall risk decreases exponentially (function $k \mapsto n(1-(1-p)^k)$ when the number of parts increases while the related cost increases accordingly. The risk decreases very fast for partitions into relatively few parts and even a small number of parts can lead to a significant decrease in the overall risk. On the contrary, partitioning the land into many parts does not provide a significant additional benefit since the risk function becomes quickly very flat as the number of parts increases. Meanwhile, in this instance, the cost of the firebreaks is nearly proportional to the number of parts.

Therefore there must be a trade-off between a low value of risk and the cost of firebreaks (installation, maintenance and land consumption). The simulation provides

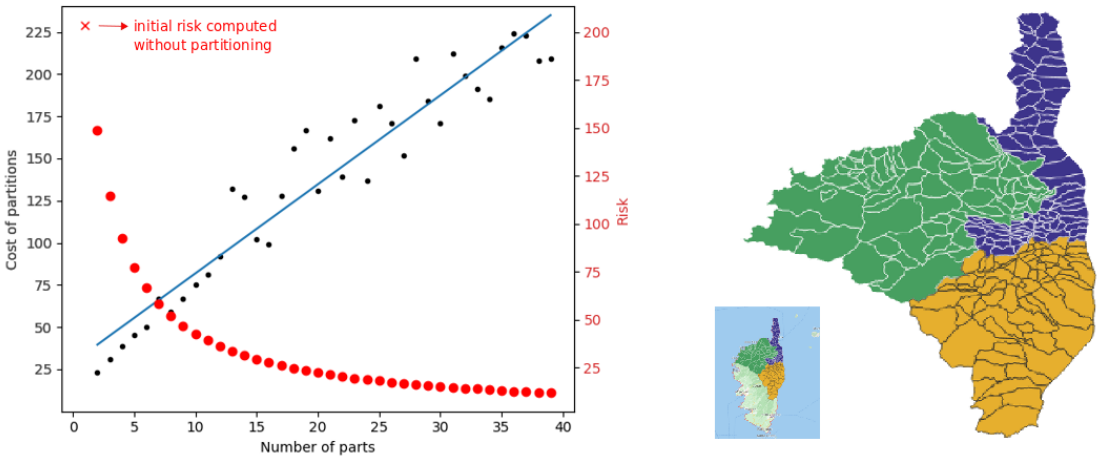


Figure 2.3: Application of a multilevel partitioning algorithm on the North of Corsica (France). On the left: the reduced risk and related cost depending on the number of parts; on the right: the result of partitioning into three parts.

also a possible operative tool for decision-making. In fact, given a budget B, plots allow to find the maximum number of achievable parts and the related reduction of risk compared to the initial state. The slope of the risk function for this budget is also a good indicator of the expected impact of increasing or reducing the budget. Note in the example the apparently erratic behavior of the cost plot that corresponds to the value of the computed partition. Without the balanced constraint, the optimal cost should always increase with the number of parts while this is not necessarily the case with balanced partitions. However, in the present case, this behavior can mostly be attributed to the error between the heuristic solution and an optimal one. The regular increase of the cost for a relatively small number of parts is an indicator of a better expected behavior of the algorithm and more interestingly, this occurs in the most interesting zone in terms of risk reduction. This illustrates the interest of the approach from an operational point of view.

Figure 2.3 shows a 3-partition for the geographical area of North of Corsica obtained with the multilevel partitioning algorithm. This example shows as well the limit of the approach using k-Graph Partition, mostly due to the underlying hypotheses. When applied on a large land, it will induce solutions with very long firebreaks that may rapidly become impossible to set-up. Meanwhile, in our example, assuming in the worst case that the full land in one part may burn neither represents a possible practical outcome, nor an acceptable one. This approach per se is more suitable for small landscapes. However, in a large and non homogeneous landscape as North of Corsica, it is still a useful indicator taking into account that natural barriers like mountain crests can act as long firebreaks at reasonable costs.

2.3 Graph model validation

In this section, we validate the graph model by applying it to Cap Corse, the peninsula to the North of Corsica Island. We estimate the model variables using data on historical fires and fire simulations. We validate the values by comparing them to the relevant characteristics of the territory, like the orography and the type of vegetation. Corsica has a high risk of fires due to climate conditions, warm and dry during summer, and due to the large extension of forests that can burn easily. Aware of these risks, firefighters, risk managers, as well as researchers in Corsica are engaged in finding solutions to face the threat of fire. Therefore there are public databases of historical fires and a web-based fire simulator, called ForeFire [70], particularly tailored to the territory. Our idea is to take advantage of this rich availability of open data and the simulator. These are the reasons why we locate our simulations in Corsica. Moreover, this graph model has been proposed and discussed with a fire agency in Corsica (France) during the European project GEO-SAFE. We firstly introduce some terminology in Section 2.3.1 and list software tools in Section 2.3.2 and the data sources in Section 2.3.3. Successively we introduce the case study and give some information about Cap Corse in Section 2.3.4. Then, we focus on the model set-up in Section 2.3.5 and explain how to identify areas in the territory, how to estimate the probabilities of ignition for each area and the probabilities of spread for each edge, and how we run the simulations in Section 2.3.6. Finally, we discuss the results.

2.3.1 Fundamental concepts

Here follows some basic terminology used in the rest of the section.

Land Cover. Land cover is a physical description of the earth's surface acquired with remotely sensed imagery. It identifies the materials which cover the ground, like grass, trees or waters leading to the classification into different categories like the type of vegetation, trees, bushes, and urban areas. A variety of image preprocessing and processing algorithms are used to map the different patterns. These algorithms detect changes at diverse spatial scales using machine learning techniques and statistical analysis.

Elevation Maps. Digital elevation model (DEM) files are a digital representation of topographic elevation in a form of a raster image. Each picture element represents a feature's elevation (Z) at its location (X and Y). Digital Elevation Models show the elevation of features like valleys, mountains, and landslides, not including vegetation or buildings.

Watersheds. A natural partitioning of the territory can be achieved through watersheds. Watersheds are pieces of land that collect rainfalls and snow melts into streams of water. These streams are then collected into water bodies, like rivers,

and eventually reach the sea but can also be absorbed by the soil. Watersheds can vary in size, as little as a lake, or extend for hundreds of square miles. Dividing the territory into watersheds is a rational strategy for defining areas related to their natural resources, as opposed to a segmentation based on administrative borders that do not consider the ecosystem or wildlife habitat. Watershed boundaries are also a natural barrier to the spread of fire, and that is why firebreaks are often located on them. Long fire barriers can be obtained by taking advantage of natural obstacles to fire diffusion, like mountain ridges. The segmentation of the land with watersheds is used, for example, by the Spanish forestry service [129]. See Figure 2.4 for reference. The size of each watershed depends on the topography of the land. It can be as little as a single stream of water, but bigger watersheds can be achieved by grouping smaller ones. The size of watersheds must be chosen carefully as a function of the processed information. The level of segmentation must be tailored to territory characteristics avoiding excessive or coarse segmentation.

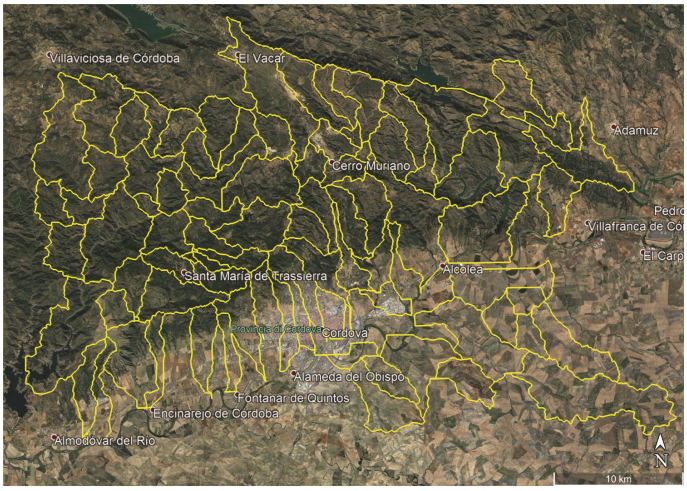


Figure 2.4: An example of the segmentation of the land by watersheds. The map is from Cordoba, Spain.

2.3.2 Software tools

In this section, we list the software tools we use for the model set-up.

Quantum Gis. (QGIS) is an open-source geographic information system (GIS) application that allows the analysis, elaboration and visualization of geospatial data.

ForeFire. is a software fire simulator [70] for the territory of Corsica. Fire simulators estimate fire propagation, given a starting point of ignition. Many simulators incorporate physical parameters like the type and moisture of fuel, topography, and weather conditions like wind, temperature, and humidity. All these parameters influence fire propagation. The simulation results can help the planning of intervention

measures to extinguish a fire. ForeFire is a software library available under the GPL open-source license. It is a discrete event simulator modeling the evolution of the fronts of fire over a wide area and at high resolution. The fire front is the contour of the burning area that expands in each of its points. ForeFire simulates a fire taking into account parameters like the temperature, wind direction, and intensity. It outputs, at each simulation step, the extent of the land affected by the fire as a GIS polygon feature. A polygon is an object that encloses an area represented as a series of x and y coordinate pairs. The simulation runs on the hypothesis of free fire propagation, without the intervention of firefighters. ForeFire is a web-server that can be queried with REST APIs. To access ForeFire, we developed a complete toolchain according to the client-server architecture. The client is a computer running a custom Python program that sends queries to ForeFire and processes the answers. Due to the complex system, it is worth checking the output of the whole toolchain in a simple test. We show the burned area in relation to different wind intensities and directions. In particular, we select two ignition points: one located in a relatively flat area and another one in a mountainous area. On the left of Figure 2.5 is shown the ignition point in yellow, chosen in a relatively flat area. In the first simulation, we set the wind blowing toward the South at four different wind intensities, 2, 4, 8, and 16 meters per second. The burned areas grow in the South direction as wind speed increases, as expected. The simulation is repeated setting wind blowing toward the North. Figure 2.5.right shows the results of the simulations. Burned areas are represented superimposed and with different colors. The darkest colors correspond to the highest wind speed. Smaller inside areas correspond to wind at a lower speed. As expected, areas grow toward the North direction and increase in dimension as wind intensity increases. The simulations are repeated for an ignition point located in the mountains. The results are shown in Figure 2.6. The simulation output behaves similarly to the previous one but, here we can see the guiding effect of the mountain crests that are not crossed by the fire.

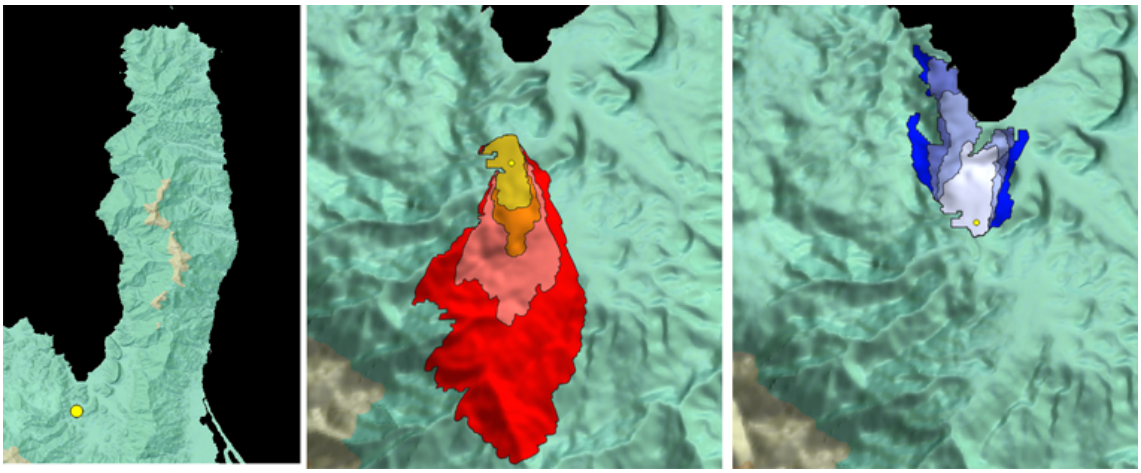


Figure 2.5: Left: the chosen ignition point in yellow on a relatively flat area. Middle: the result of simulations with the wind blowing South. Right: results of simulations with the wind blowing North. Simulations are carried out with increasing wind speeds.

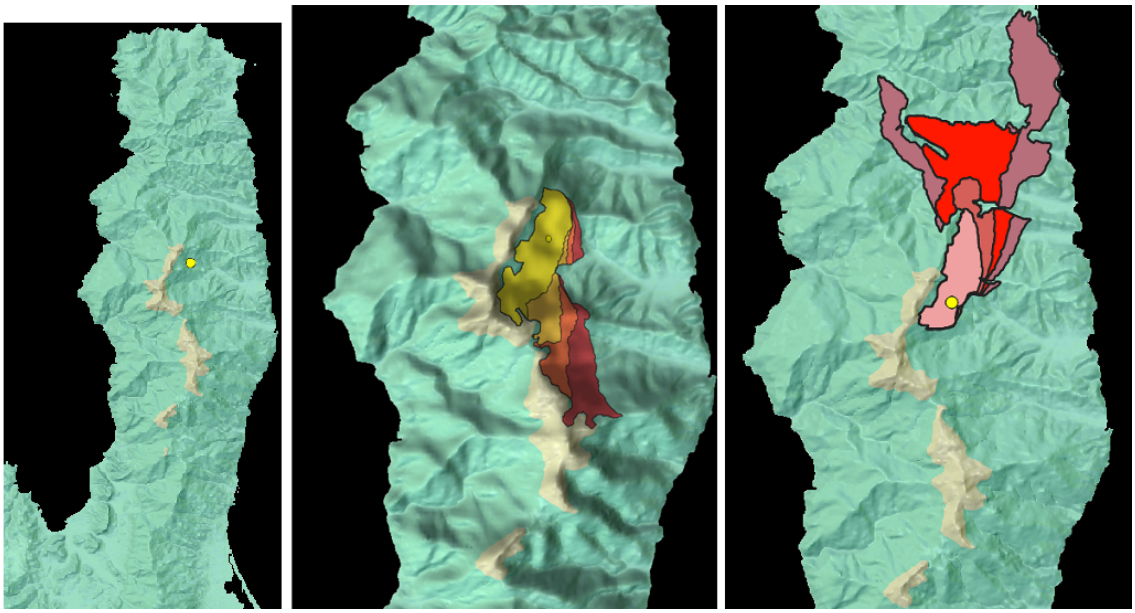


Figure 2.6: Left: the chosen ignition point in yellow on a mountainous area. Middle: polygons resulting from simulations with the wind blowing South. Right: results of simulations with the wind blowing North. Simulations are carried out with increasing wind speeds.

2.3.3 Data sources

In this section, we list the data used for the model set-up and the subsequent analysis.

We use DEM files of Corsica from Copernicus [66] European project with a resolution of 25 meters, sourced from Open-Dem website [10].

Data on historical fires that occurred in Corsica from the year 1973 to 2020 come from Promethee [116], a public Forest fires database for the Mediterranean area in France. Every entry reports the administrative area from which a fire ignited and an approximate location in the DFCI grid which is a geographical grid system used in France by the actors of the Defense of Forests Against Fires (DFCI). The territory is divided into a square grid. Each square has a side of 100 km and it's further divided into smaller squares until squares have a side of 2 km. We assigned a fire to the nearest centroid of the DFCI grid. For 38% of the data is also available a description of the cause of the fire that is, arson in 35% of the cases.

We use ESRI land cover maps derived from ESA Sentinel-2 imagery at 10m resolution [91]. Maps provide a classification of the surface of the earth, including vegetation types, bare surface, water, cropland, and built areas. We list the class definitions used in [91] and relevant for our analysis.

- **Trees**: any significant clustering of tall (15 feet or higher) dense vegetation, typically with a closed or dense canopy; examples: wooded vegetation, clusters of dense tall vegetation within savannas, plantations, swamp or mangroves (dense/tall vegetation with ephemeral water or canopy too thick to detect water underneath).

- **Flooded vegetation** Areas of any type of vegetation with obvious intermixing of water throughout a majority of the year; seasonally flooded area that is a mix of grass/shrub/trees/bare ground; examples: flooded mangroves, emergent vegetation, rice paddies and other heavily irrigated and inundated agriculture.

- **Crops** Human planted/plotted cereals, grasses, and crops not at tree height; examples: corn, wheat, soy, fallow plots of structured land.

- **Built-area** human made structures; major road and rail networks; large homogeneous impervious surfaces including parking structures, office buildings and residential housing; examples: houses, dense villages/towns/cities, paved roads, asphalt.

- **Bare-ground** Areas of rock or soil with very sparse to no vegetation for the entire year; large areas of sand and deserts with no to little vegetation; examples: exposed rock or soil, desert and sand dunes, dry salt flats/pans, dried lake beds, mines.

- **Range-land** open areas covered by homogenous grasses with little to no taller vegetation; wild cereals and grasses with no obvious human plotting (i.e., not a plotted field); examples: natural meadows and fields with sparse to no tree cover, open savanna with few to no trees, parks/golf courses/lawns, pastures. Mix of small clusters of plants or single plants dispersed on a landscape that shows exposed soil or rock; scrub-filled clearings within dense forests that are clearly not taller than trees; examples: moderate to sparse cover of bushes, shrubs and tufts of grass, savannas with very sparse grasses, trees or other plants.

2.3.4 Cap Corse in Corsica: a case study

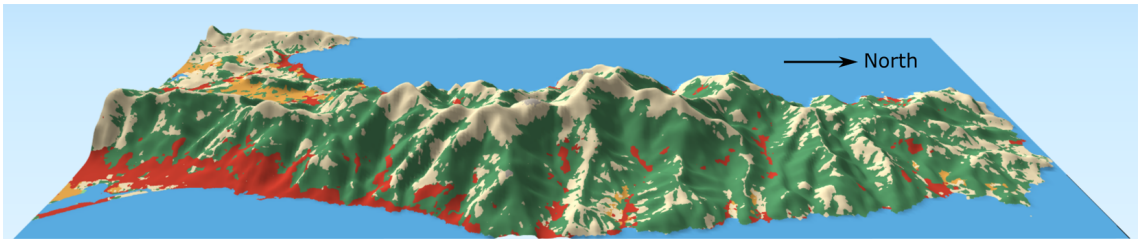


Figure 2.7: A 3d elevation model of the peninsula of Cap Corse, Corsica.

Cap Corse is a peninsula 40 kilometers long and 10 to 15 kilometers wide located in the North of Corsica. The Serra mountain range extends across the length of the peninsula, from the Serra di Pignu (960 m) in the north to the mountain Castellu (540 m) in the south. The highest peak is 1324 m tall, but also other peaks exceed 1000 m; the coasts are very steep.

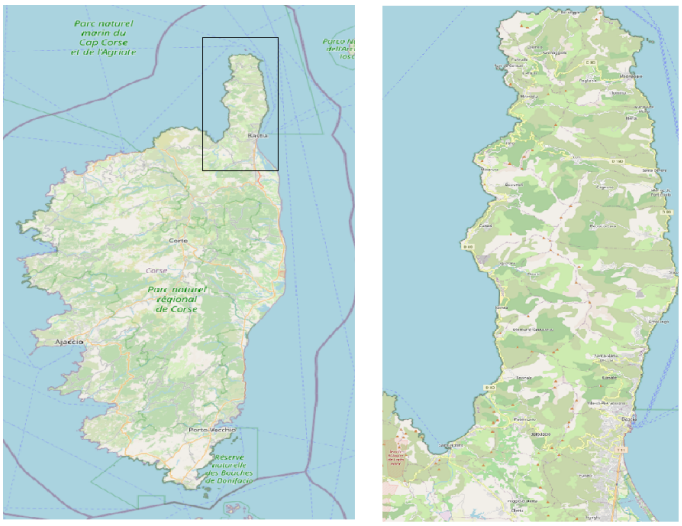


Figure 2.8: The peninsula of Cap Corse, located at the North of Corsica.

The vegetation is mostly Mediterranean scrub up to 900 m. At higher heights, we find broad-leaved trees, mostly chestnut, and coniferous trees. Analyzing the ESRI land cover map [91] and according to the identified classes, we found that the 57% of the land is covered by trees, 32,3% is range-land, 2,6% is crops, and 2,8% is built area.

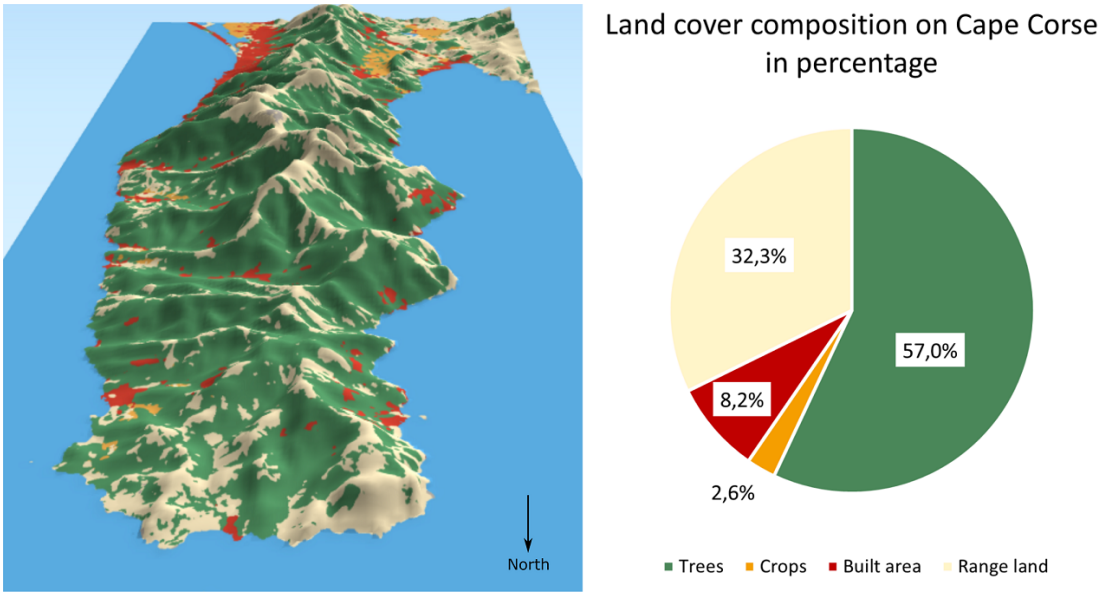


Figure 2.9: On the left, a 3d elevation model of the peninsula of Cap Corse, colors represent land cover, on the right the distribution of land classes in percentage.

2.3.5 The graph model set-up

In this section, we delve into the model set-up.

Vertex. As a first step, we partition the land into watershed areas, each of them representing a vertex v in our graph. We use DEM (Digital Elevation Model) maps and Quantum GIS (QGIS) [119] to compute the extension of watersheds. We partition the territory of Cap Corse into 34 watersheds and we assign an id to each of them. See Figure 2.10 for reference. To estimate a value $\varphi(v)$ for each vertex, we should take into consideration the main characteristics of the area like the type of fuel and the presence of valuable assets like for example forests or cities. This estimation requires the involvement of both wildfire managers and administrative people. As a first approximation, we estimate the value of each vertex as proportional to the area.

The probabilities of ignition associated to each vertex of the graph are evaluated using data on historical fires from Promethee [116]. We assigned a fire to the nearest centroid of the DFCI grid. To estimate the probabilities of ignition, we count the number of fires that hit a watershed. Given a node $v \in V$ and the set of all recorded historical fires F_v that hit the area identified by node v, the probability of ignition for v is estimated as $\pi_i(v) = \frac{|F_v|}{|F|}$, where F is the set of all recorded fires.

Table 2.1 reports the estimated probabilities of ignition $\pi_i(v)$ and in Figure 2.11 the probabilities are embedded on the map. In Figure [46], the ignition probabilities are

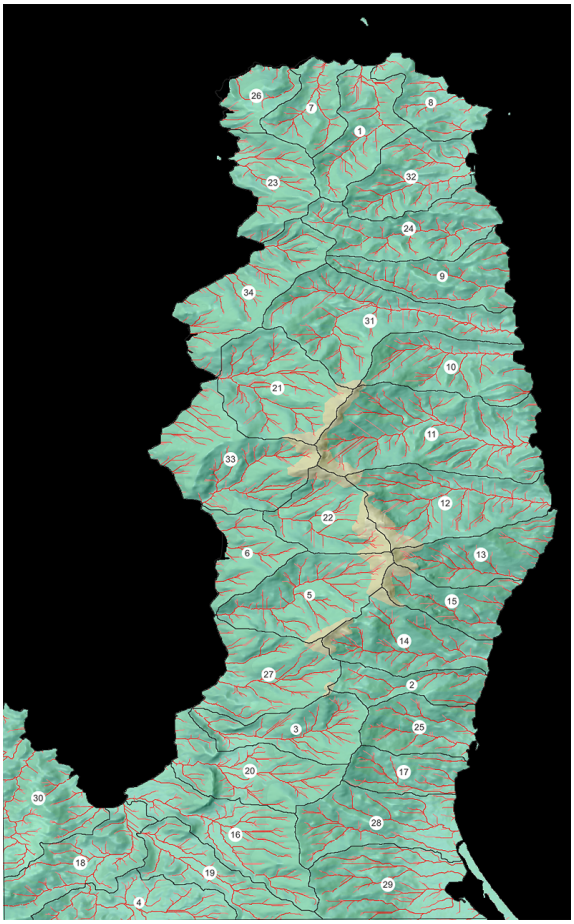


Figure 2.10: An elevation map of Cap Corse partitioned into 34 watersheds. Numbers corresponds to watershed ids, red lines are water streams.

represented visually with color tones varying from white to red. White corresponds to the lowest value of probability, and the darkest tones of red to the highest value of probability. Only 38% of the data on historical fires report a classification by the cause of the fire, and 35% of fires were classified as arson. Figure 2.12, shows a close-up of the map where the probabilities of ignition are higher over the land cover map. Confronting the values of the ignition probability with the land cover map, we can see that the areas with the highest probability of ignition contain urbanized and cultivated areas.

Edge. To estimate the probabilities of spread, we use ForeFire [70] to simulate a fire from an ignition point. We then analyze the spread of fire on the neighbor watersheds.

We choose the points of ignition (samples) uniformly at random. Each simulation outputs an area burnt by the fire, encoded in JSON format [88]. Figure 2.13 shows an example of the result of a simulation. A point of ignition is represented in yellow

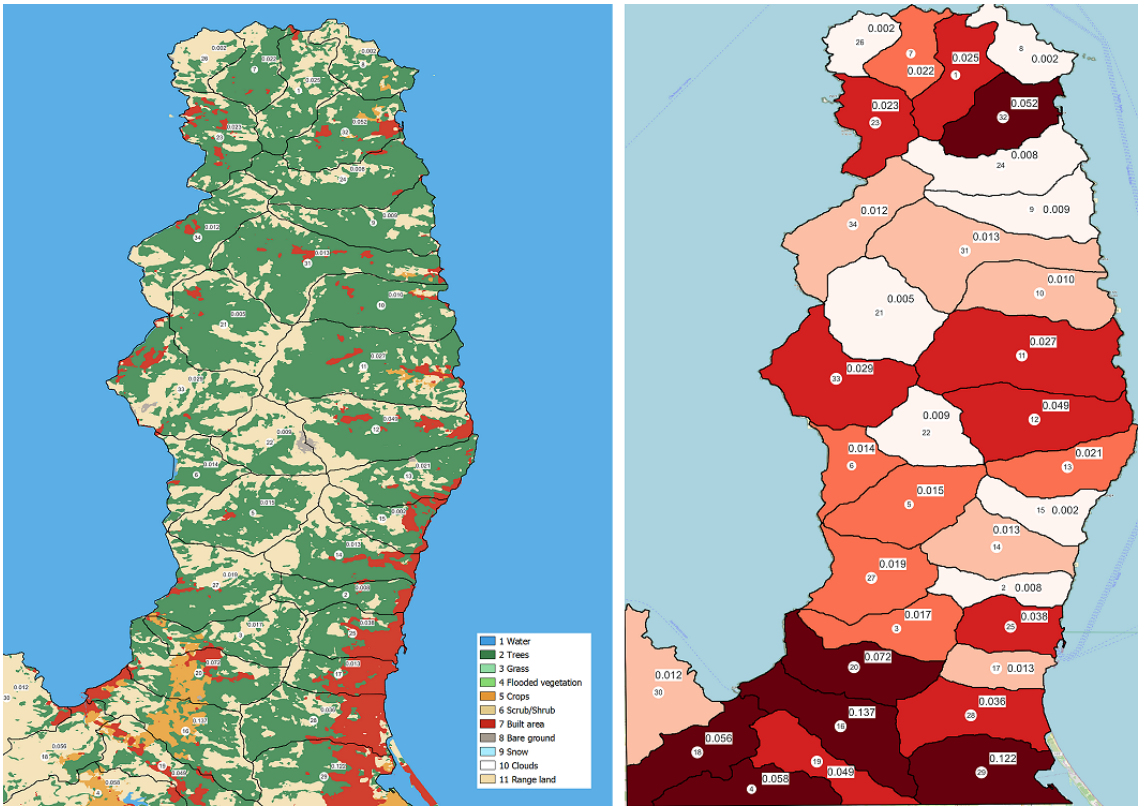


Figure 2.11: The estimated probabilities of ignition reported on the map. Numbers into circles correspond to catchment ids. Numbers inside rectangles are the estimated probabilities of ignition. On the left: land cover; on the right: a visual representation of the probabilities values in tones of red. Dark red corresponds to the highest value.

while the red area is a polygon that represents the extension of the fire in the simulated time. We intersect the area resulting from the simulation with the map of watersheds to compute the number of times in which a fire ignited in a watershed, spreads in one of its adjacent neighbors. Even if the fire spreads to a greater distance (for example, two hops neighbors), we observe only the one-hop neighbors of the watershed. In Figure 2.13, a fire ignites in watershed 34 and expands in the North direction in six other watersheds: 9, 24, 32, 1, 7, 26. Only 9, 24, and 23 are one-hop neighbors, others are at a greater distance.

To estimate the spread probability from watershed w_i to watershed w_j, we collect the number of X_{ij} positive cases in which a fire spreads from w_i to w_j in a group of n_i simulations from n_i ignition points. X_{ij} is a binomial random variable and the sample proportion p_{ij} can be expressed as follows: $p_{ij} = X_{ij}/n_i$. The variance of p_{ij} is $\frac{p_{ij} \cdot (1-p_{ij})}{n_i}$ and the standard error equals the square root of the variance. Therefore

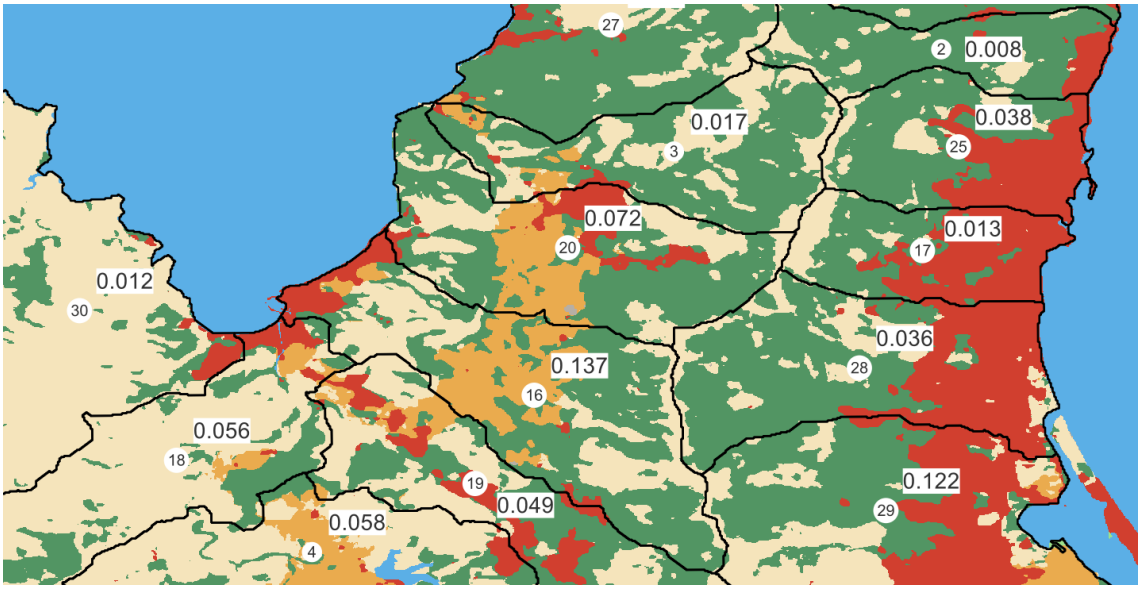


Figure 2.12: The map shows the areas with the highest probabilities of ignition.

the standard error is computed as follows:

$$\sqrt{\frac{p_{ij} \cdot (1 - p_{ij})}{n_i}}$$

As the size of the sample increases, the standard error decreases, and - as a consequence - the precision in the estimation of the spread probability increases. Accordingly, we add ignition points until the standard error becomes - for each watershed - less than a defined threshold, or we reach a maximum number of iterations. In our simulations, the threshold is 0.05, and the number of iterations varies between 10 and 50. The estimation of the cost for each edge, that corresponds to the installation of a firebreak requires the involvement of many stakeholders and depends on the orography of the territory, access ways, and other parameters that must be taken into account. As a first approximation, we evaluate the cost of a firebreak installation as proportional to the length of the boundary shared by two neighbor watersheds.

v	$\pi_i(v)$	v	$\pi_i(v)$
1	0,025	18	0,056
2	0,008	19	0,049
3	0,017	20	0,072
4	0,058	21	0,005
5	0,015	22	0,009
6	0,014	23	0,023
7	0,022	24	0,008
8	0,002	25	0,038
9	0,009	26	0,002
10	0,010	27	0,019
11	0,027	28	0,036
12	0,049	29	0,122
13	0,021	30	0,012
14	0,013	31	0,013
15	0,002	32	0,052
16	0,137	33	0,029
17	0,013	34	0,012

Table 2.1: The table shows the estimated probabilities of ignition $\pi_i(v)$ for the 34 watersheds. v is the watershed id.

2.3.6 Simulations and results

The simulation toolchain comprises a local client and the remote server ForeFire. The local client has been instructed to query the remote server with a fixed wind direction blowing from North-West to South-East. This is the dominant wind direction during summer in Corsica as told by fire agencies.

The ignition points are chosen uniformly at random for each watershed. The simulation proceeds in time steps, each one representing 20 minutes of simulated time. We set up a simulated time of 6 hours of free fire propagation before the intervention of the firefighting organizations. The local client collects the resulting polygons and computes the spread graph as explained in the previous paragraph. Figure 2.15 shows the resulting spread graph. Each watershed is identified by an id while each edge has an associated probability of spread. We notice that probabilities of spread are higher on edges pointing to neighbors located relative South rather than those located relative North, in accordance with the wind direction that we set. It is evident that the mountain range crossing the peninsula acts as a natural firebreak blocking the spread of fires across them. Figure 2.14 is a close-up of the spread graph, represented over the relief map. Examples of edges with a zero probability of spread crossing the mountain range are $(11, 21)$, $(33, 11)$, $(11, 33)$, $(22, 11)$, $(11, 22)$, $(12, 22)$, $(13, 22)$, $(5, 13)$, $(13, 5)$, $(5, 15)$. Fires ignited in one of these watersheds do not spread over the neighbors located beyond the mountain range. Figure 2.15.right

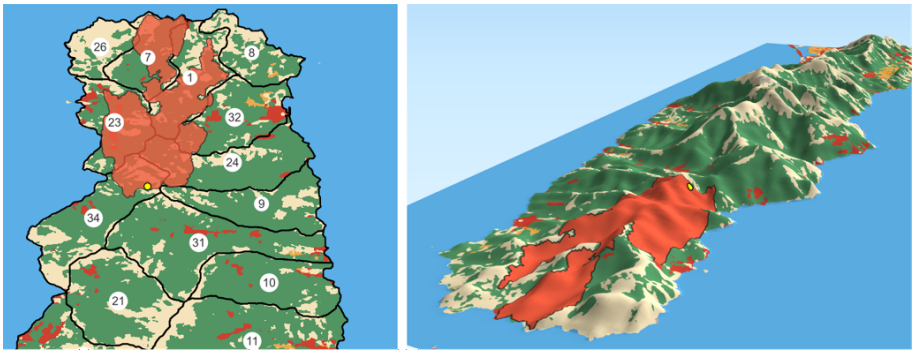


Figure 2.13: On the left a 2d view of the outcome of a simulation. On the right the corresponding 3d view. The point of ignition is represented in yellow, the surface in red represents the burnt area.

shows ignition points over the land cover map. Points are color-coded with tones of red to represent the total extension of each fire resulting from the simulations, expressed in hectares.

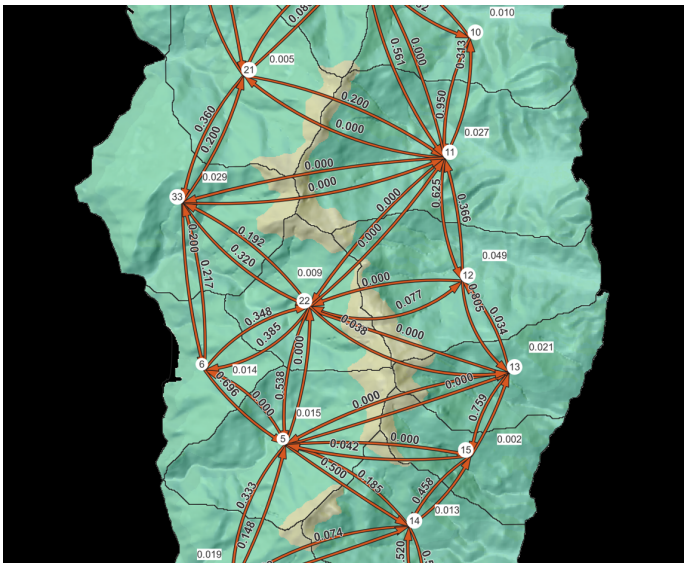


Figure 2.14: A close-up of the graph resulting from the simulations over the relief map. Mountain Crests block fire spreading.

Figure 2.15.right, shows big fires in the valleys on the east side of the peninsula covered by forests. Fires ignited on coasts and covered by a range-land type of fuel, often result in small fires either because coasts are steep or because the fuel is made of homogeneous grasses with no taller vegetation. A fire ignited on the top of mountains has a small extension. Indeed mountains around 1000 m high, do not have tall trees on the peaks. Medium-size fires are located in the South-West of Cap Corse in which the fuel consists of cultivated fields, while smaller fires are on the

South-East Coasts in highly urbanized areas where the natural vegetation is low. In this simulation data are coherent with the fuel type and with the wind direction. These simulations are a proof of concept to show a method to estimate the model elements like the probabilities of ignition and the probabilities of spread, and how to define the watersheds. Further evaluations are needed to estimate the value for each area and estimate the cost for each firebreak. The simulation confirms that wind intensity and direction have a large impact on the spread graph. Therefore it is reasonable to run simulations with different wind conditions according to seasonal changes.

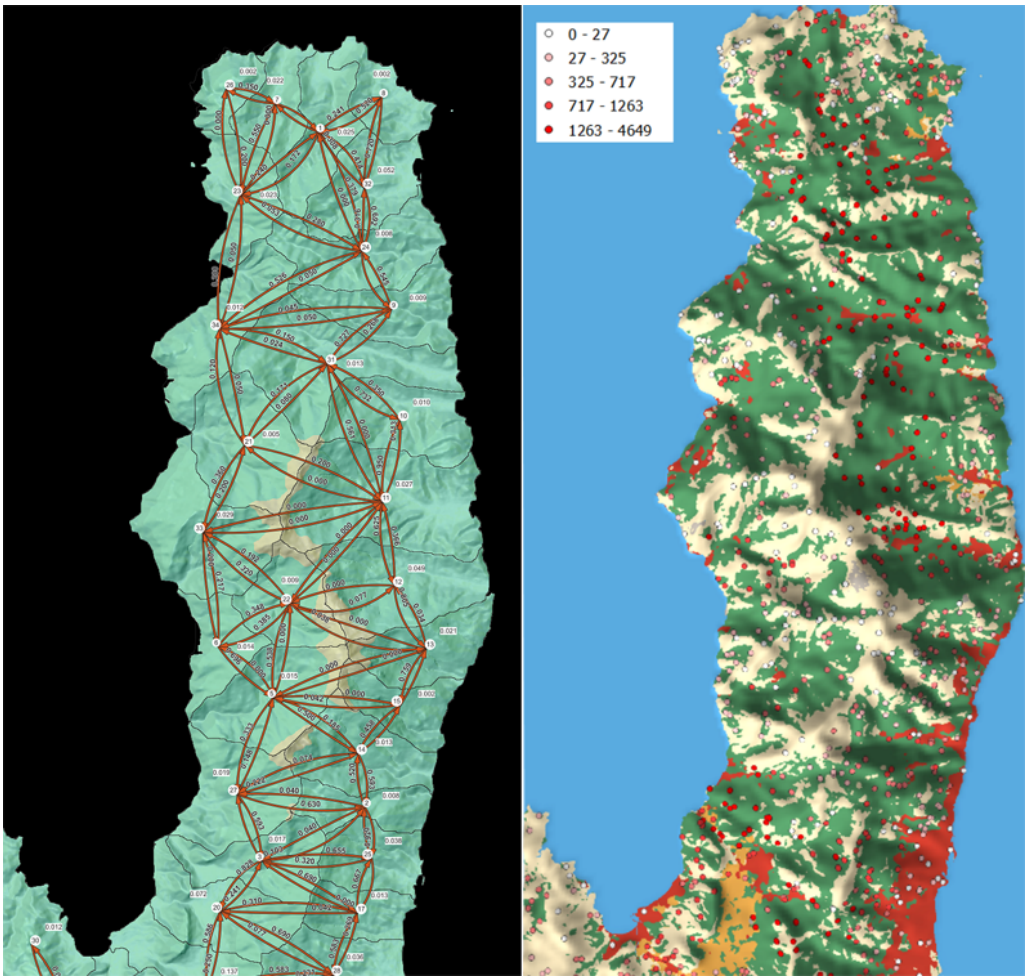


Figure 2.15: On the left: the spread graph resulting from the simulations. Labels inside circles are watershed ids, labels inside rectangles are estimated ignition probabilities related to watersheds, and labels on edges are the estimated spread probabilities. On the right: the ignition points over the land cover map. Points represent, in tones of red, the extent of the burned area in ha.

2.4 The web application

The web application is a prototype, we are developing to support decision-makers and stakeholders to design wildfire preventive measures. The application shows the data through risk maps and allows the user to interact with built-in algorithms in a user-friendly interface. The application implements algorithms to compute the burning probabilities and the risk, to solve the firebreak location problem using a greedy approach described in Section 2.1 as well as the partitioning problem. The system includes a relational database [94] to store data while interactive maps are elaborated using the Leaflet library [12].

2.4.1 The interface

The interface presents the available functionalities in a menu bar at the top. Each option corresponds to a visualization of a different kind of wildfire data, represented on geographical maps. The "Ignition probability map" shows the ignition probabilities on the map of Cap Corse. Figure 2.16 shows the territory divided into watersheds. Each watershed is colored with a shade of red, from white to dark red, meaning low to high probabilities. A legend at the bottom shows the color-scale correspondence. Once a watershed is selected (or hovered with the mouse), a pop-up menu visualizes the value of the probability of ignition.

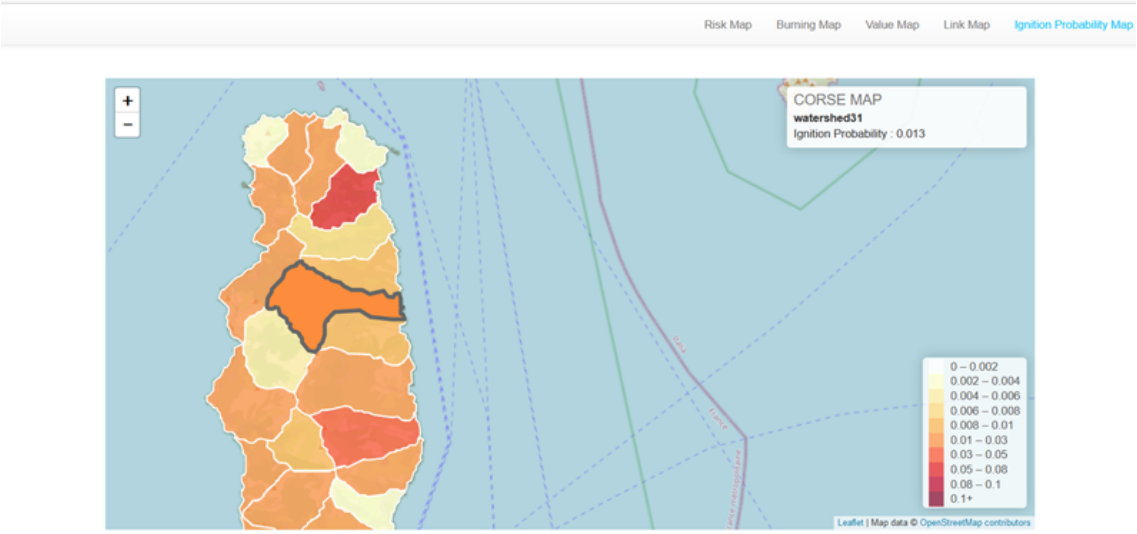


Figure 2.16: A screenshot of the application. A view of the "Ignition probability map".

The "Value map" allows the user to click over a watershed and visualize the associated value.

The "Link map" shows the adjacency between the watersheds with a directed graph. When a link is clicked, a pop-up menu shows the starting watershed, the end watershed and the associated probability of spread. See Figure 2.17 for reference.

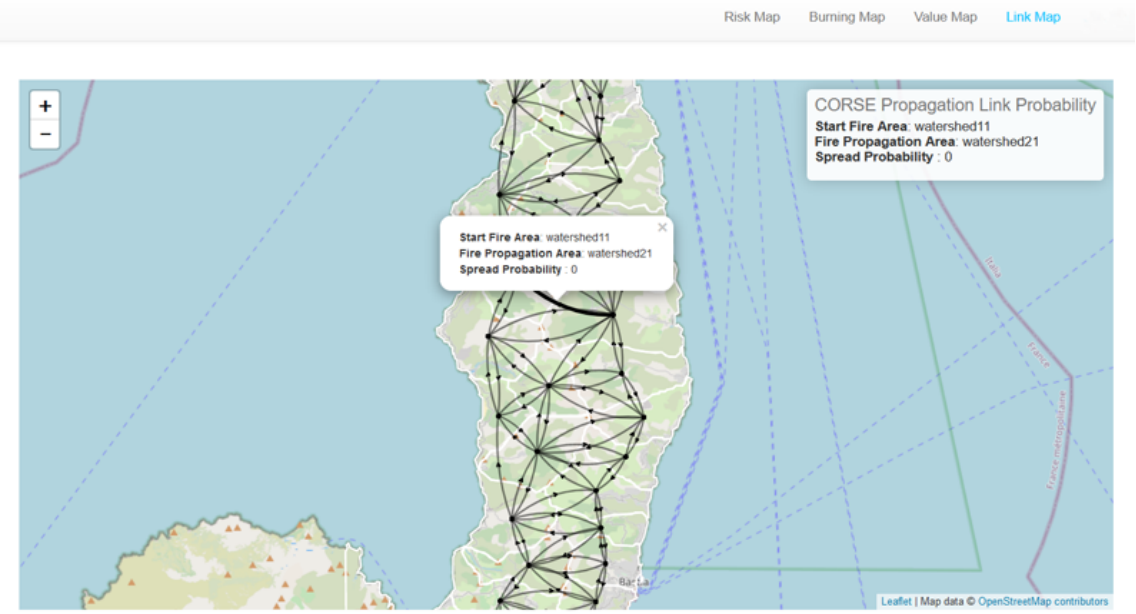


Figure 2.17: A screenshot of the application. A view of the "Link map".

By clicking on the "Risk map" menu option, the web app invokes the engine to compute the risk using the algorithm described in Section 2.1. At the end of the calculation, each watershed is colored in increasing tones of red (from the lower to the higher risk, respectively). The application implements also the algorithm to address the partitioning problem using the multi-level partitioning algorithm when the probabilities of ignition are equal and all the vertices have the same value as described in Section 2.2.5. The web app also enables registered users to evaluate what-if scenarios. In particular, fire managers can simulate the installation of a firebreak and see the corresponding reduction of the risk. Once an edge is selected, the popup window that opens on the top-right of the map allows you to either edit the related information or remove the selected edge. To implement this functionality, the web app keeps a copy of the original graph and compares the effects of any change with the original one. The removal of both directed edges between two watersheds corresponds to the installation of a firebreak. Once done, by clicking on the "Risk reduction map" option, the risk map for the original and modified graphs are computed. Then, the web app shows graphically the simulated risk reduction with green tones, the darker the green, the higher the risk reduction. Figure 2.18 shows the interface of the "Risk reduction map" after the removal of a few edges from the graph, the map shows the corresponding risk reduction.

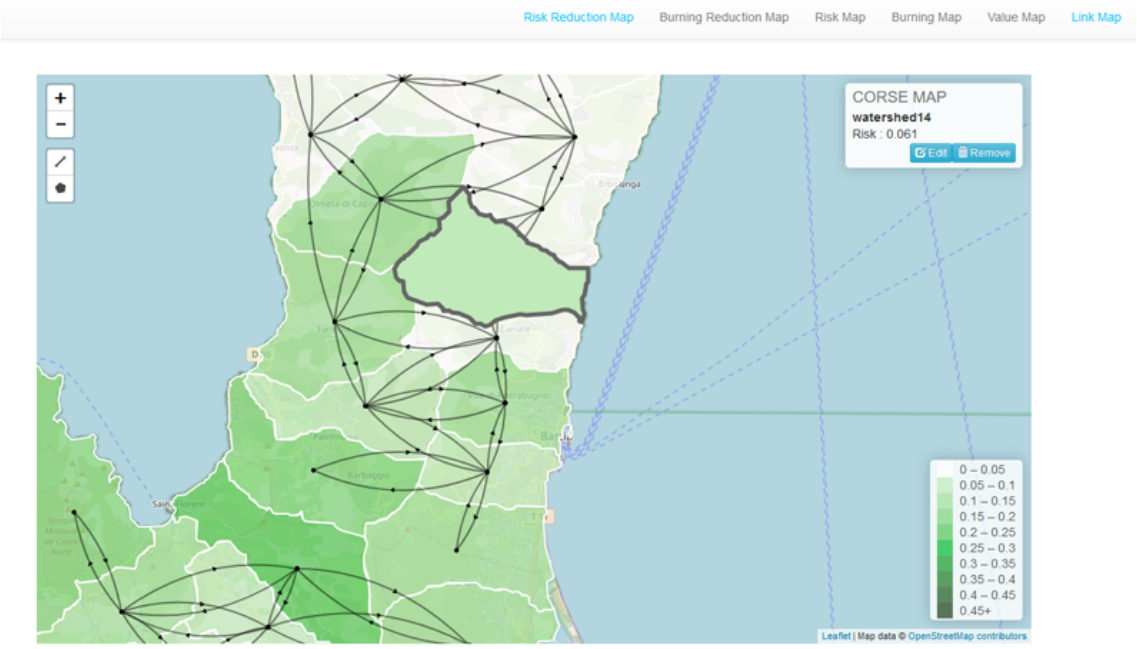


Figure 2.18: A screenshot of the application. A view of the "Risk reduction map".

2.5 Concluding remarks

In this chapter, we studied heuristic approaches for the Firebreak Location problem relevant to practical applications. We showed a heuristic algorithm to compute the risk and we presented a variation of Firebreak Location in which a uniform territory is partitioned in subareas of approximately the same dimension using firebreaks. We tackled the problem with multilevel graph partitioning and showed that separating the territory even into a few partitions may lead to a significant fire risk reduction.

We validated the modeling of a landscape as a graph by applying it to the territory of Cap Corse. We estimated the probabilities of ignition with data about historical fires and estimated the probabilities of spread using a software fire simulator. This case study constitutes a proof of concept that shows how the model can be applied practically. The results of the simulation are in accordance with the wind direction and fuel type and confirm that wind intensity and direction have a large impact on the spread graph, therefore it is reasonable to run simulations with different wind conditions according to seasonal changes. Finally, we presented a prototype of a web application that enables users to instantiate, visualize and modify the underlying graph, calculate the fire risk, and simulate the effect of the application of preventive measures like firebreaks in an easy-to-use interface.

Part II

Models and algorithms for distributed robots

Outline

Robot technology has advanced quickly, assisting humans in increasingly complicated tasks. The development of robots able to operate in forest environments can help in tasks like firefighting and fire prevention by land monitoring. For example, in a scenario of a forest fire ignition, a timely intervention can avoid an uncontrolled spread of fire and the loss of acres of vegetation. Drones can monitor the evolution of the front of a fire with onboard infrared and visual cameras. Multiple drones can cover big areas and obtain supplementary views of a fire scene [107]. In a search and rescue scenario, robots can inspect the scene of a fire and locate survivors [103, 122]. After the wildfire, robots can collect data about the damage caused by the fire.

These robots, equipped with tools to extinguish a fire, can be the first line of defense by protecting the lives of firefighters and helping them to reach areas that are not accessible or hazardous for humans; when equipped with smoke sensors and thermal cameras, robots can locate a fire better than humans. Robots are employable in forest environments [112] for tasks like wildfire firefighting [121], data collection [132], fire monitoring [107], and forest pruning [111]. A multi-robot system was developed to detect combustible materials and remove them [47]. In [82] drones, equipped with thermal cameras, perform the tracking of wildlife in forests or open areas.

These are very challenging tasks for a robot, different from one another. In forests, robots move in unstructured contexts, with limited bandwidth and poor visibility. Wildfires make the task even more dangerous. A robot can be damaged while doing its job or its task can change over time. Will this robot be adaptable or will it be necessary to re-design it? Two opposite robot design approaches are single-purpose robots and swarming robots. In the first case, a specialized robot solves a specific task. Mobile robots can be very complex systems that require the integration of different expertise to keep them working. An alternative design approach is to use simple robots, much less specialized in their features but, able to cooperate to do a task. Each robot could have sensors, a microcontroller, batteries, and means of locomotion. Robots act as a team to do their task by implementing collective behavior. Swarm robotics draws inspiration from biological systems. Many animals show a collective behavior: migrating birds that flock in formation, bees colonies, and fishes that shoal together so there are fewer chances to be captured by predators. In all these cases, a group is made of similar individuals that behave autonomously,

implement simple actions, and interact with the environment. The group, taken as a whole, shows intelligent behavior. Animals cooperate to reach goals like searching for food or defending themselves from predators. The single animal is not able to finalize these goals by himself, the group instead, solves difficult problems. The goal of swarm robotics is to design robots so that they can cooperate to do different tasks. There are many potential advantages of a swarm of robots compared to single-purpose robots. Natural swarms show some desirable features for artificial systems, like versatility, scalability, and robustness. The group of robots is made of simple, identical individuals, that can be cheap and can be mass-produced. If one fails, it can be easily replaced, more difficult is instead to repair a broken part of a stand-alone robot. These robots are autonomous and act independently: the system is thus decentralized. There is no central entity giving directions to robots. In fact, as the number of entities increases, centralized control cannot be efficient or scalable. The coordination of a multi-robot system in risky firefighting contexts cannot rely on direct communication between robots or with a central node because requires sharing information in real-time and with limited bandwidth. Therefore autonomous swarms, in which robots do not communicate directly could be more efficient [16]. Many studies focus on the design of multi-robot systems employed in fire-fighting tasks [11, 19, 86, 87]. The distributed computing paradigm offers a reliable solution that allows the designing of a system that self-organizes and adapts to environmental changes. However, the coordination of distributed group robots poses big design challenges. The first challenge is to adopt a theoretical framework in which the main features can be modeled and studied. The second challenge is posed by the design of distributed algorithms taking into account robots' abilities and deciding if a task is feasible. The third challenge is to deploy algorithms on physical robots with all the problems arising in terms of choice of robots, protocols, reliability, cyber-security, interface with humans, and so on. The advantage of a theoretical approach is the opportunity to focus on the algorithmic issues that are difficult to address if the research starts from experimentation with robots. In this thesis, we follow a theoretical approach and we focus on the design of algorithms for distributed robots under the hypothesis of the $\mathcal{OBLOT}$ [73] model in which robots are modeled as distributed agents. Like in a swarm, $\mathcal{OBLOT}$ robots have minimal capabilities, do not communicate directly and are able to move in some environments like the Euclidean plane or discreet spaces like grid graphs. Grids can be interpreted as a discretization of the plane in which the movements of the robots are quantized and the system has a unit of measurement. A territory can be divided into areas and robots move from one area to another. If areas are squares and adjacent the result is a grid, otherwise, in general, is a graph.

In Chapter 3 we recall the main features of the $\mathcal{OBLOT}$ model in a cohesive section as it is the underlying model of the algorithms presented on graphs in Chapter 3 and the model that we extend in Chapter 4. We then present the solution to different variations of the pattern formation problem that asks to a group of robots to reach final positions according to a geometric shape given in input. Patterns can be given

in input to robots in many ways, such as a set of points in an ideal coordinate system, as a set of composition rules in terms of relative positions to other robots, or in terms of a property that the robot configuration must meet. We present a distributed algorithm that solves the arbitrary pattern formation for robots moving on tessellation graphs, and a distributed algorithm to solve the geodesic mutual visibility problem. This problem asks to place robots so that they are geodesic mutually visible: each couple of robots has a shortest path in which no other robot resides. The study is motivated by the fact that mutual visible robots can reach any other robot along a shortest path without collision. We present the first results we achieved for robots disposed on the vertices of a tree.

In Chapter 4 we introduce $\mathcal{MOBLOT}$ a novel model in theoretical swarm robotics in which robots cluster to form complex computational units, called molecular robots inspired by the chemical paradigm in which atoms combine to make molecules. Once clustered, these robots move in a coordinated way as a new computational entity. $\mathcal{MOBLOT}$ is an extension of the $\mathcal{OBLOT}$ model and allows us to model a swarm of robots that can be divided into subgroups. Furthermore, we present the matter formation problem in which robots use a hierarchical approach to solve the pattern formation problem. We present a case study and then, we apply the $\mathcal{MOBLOT}$ model to robots moving on the square grid graphs.

Chapter 3

Robot pattern formation under the Oblot model

Pattern formation is a fundamental problem in swarm robotics. The task calls for a distributed algorithm that guides a group of robots to reach final positions according to a geometric shape given in input. In this chapter, we present distributed algorithms for the solution of two different pattern formation problems.

We adopt for robots the well-investigated $\mathcal{OBLOT}$ model [73] in which robots are equipped with very limited capabilities. Robots do not have memory and are anonymous, and autonomous, without the ability to communicate directly. The rationale behind this choice is to understand what are the minimum abilities required for robots to solve a task and to design a system that is resilient and robust. When robots do not communicate directly but draw information from the context, there is no way to hack the communication protocol. If a robot does not use its working memory and it switches off, due to temporary malfunction, it can recompute the information about the state of the system just by looking at the surrounding environment. Autonomous robots can make decisions on their own; with no centralized control, there is no single point of failure. A lot of swarm robotics tasks can be modeled under $\mathcal{OBLOT}$ like the flocking, the patrolling, the gathering [45] in which robots need to choose a location and meet there. This problem has been investigated during the GEO-SAFE project [34, 41] and can model a group of drones that identifies a fire scene and gather at that point.

The Pattern Formation(PF) has been extensively studied under the $\mathcal{OBLOT}$ model [4, 33, 37, 40, 72, 115, 131]. Given a team of robots R and a geometric pattern F expressed in terms of (a multi set of) points in an ideal coordinate system, the goal is to design a distributed algorithm $\mathcal{A}$ that guides robots to form the pattern, if possible. As usually robots do not have access to a global coordinate system, a pattern is declared formed as soon as robots are disposed *similarly* to the input pattern, that is regardless of translations, rotations, reflections and uniform scaling.

In the following Section 3.1, we recall the fundamental features of the $\mathcal{OBLOT}$ model used as the underlying model in the presented algorithms. Then, in Section 3.2 we explain the methodology used to design the distributed algorithms. We follow a decomposition approach introduced in [44]. Section 3.3 presents a resolution algorithm for the arbitrary pattern formation problem for robots moving on regular tessellation graphs. Section 3.4 presents a resolution algorithm for the geodesic mutual visibility problem for robots moving on trees.

3.1 The Oblot model

A robot with minimal capabilities is the reference model for research in theoretical swarm robotics. As mentioned, one well investigated model is certainly $\mathcal{OBLOT}$ [73]. We model the distributed agents that move on some environments according to $\mathcal{OBLOT}$. In this section, we describe its main features. A robotic system within $\mathcal{OBLOT}$ is represented by a set $R = \{r_1, r_2, \ldots, r_n\}$ of n entities, called robots, that live and operate in a connected spatial universe $\mathcal{U} \subseteq \mathbb{R}^d$, $d \geq 1$, in which they can move. Robots are considered to be **dimensionless**, i.e. as points in $\mathbb{R}^d$. This hypothesis corresponds to considering the extension of a robot as negligible for the space in which it moves. When two robots occupy the same location at the same time, we say a **multiplicity** occurs.

Robots are considered equipped with minimal capabilities.

- **identical**: they are indistinguishable by their external appearance;
- **anonymous**: they are not identifiable with an id during the computation;
- **autonomous**: there is no centralized control or external supervision;
- **homogeneous**: they all run the same deterministic algorithm;
- **silent**: they do not communicate directly with other robots;
- **oblivious**: they have no memory of past events;
- **disoriented**: each robot has its own local coordinate system - LCS, whose origin always coincides with the robot's position; the coordinate systems of different robots might have all different orientations, unit of length and handedness.

A robot has a visibility range ν, equal for all robots. It measures how far robots can observe $\mathcal{U}$. They see other robots in within the visibility range determining their positions expressed in its own LCS. The visibility is said to be **limited** if $\nu \neq \infty$, **unlimited** otherwise. A robot is equipped with **multiplicity detection** if it can detect if a point is occupied by one, or more than one robot; the multiplicity

detection is said to be strong if it allows to detect the exact number of robots on the same point, weak otherwise.

Communication. The communication between distributed agents can be of two types. When agents communicate directly (or explicitly) there is a communication channel through which agents send and receive messages to other agents. Communication can be also indirect or implicit when is mediated by the environment. Agents do not share messages but they acquire information from the environment. This type of communication is called stigmergic in swarm robotics, and it mimics the behavior of certain animals that, intentionally or not, leave a sign of their passage in the environment. That information is then exploited by others. Examples are animal footprint marks left on the ground or the traces of pheromones that ants leave going back to the nest to signal a route to a source of food or the waggle dance performed by bees to indicate the direction and distance to patches of flowers yielding to nectar. In $\mathcal{OBLOT}$ robots interact only by observing the environment and having a view of the position of all the other robots. Therefore the communication is *stigmergic*.

Each robot behaves according to a sequence of four states: `Wait`, `Look`, `Compute`, and `Move`.

- `Look`. The robot activates its sensors and observes $\mathcal{U}$ within its visibility range ν and gets a snapshot of the positions of all other robots, and their coordinates relative to its own local LCS. Since each robot is subject to a local coordinate system, it is based on relative angles and positions of robots. We will call the **view** of a robot the data structure containing all the information deductible by a robot during its `Look` phase.

- `Compute`. The robot performs a local computation according to a deterministic algorithm $\mathcal{A}$ (we also say that the robot executes $\mathcal{A}$). The algorithm is the same for all robots, and the result of the `Compute` phase is a destination point along with a path to reach it.

- `Move`. The robot goes to the computed path; if the destination is the current location, the robot stays still, performing a *nil* movement.

- `Wait`. When a robot is in wait we say it is inactive.

Robots repeat indefinitely these four steps. Such states form a **computational cycle** of a robot. The movements of the robots make the system evolve toward the next state.

Robots are oblivious: when a cycle ends they forget everything. When they wake up from the wait state, it's like the first time they are active because they have no access to information acquired in past cycles. A robot can rely only on the information captured during the look phase to understand the state of advancement

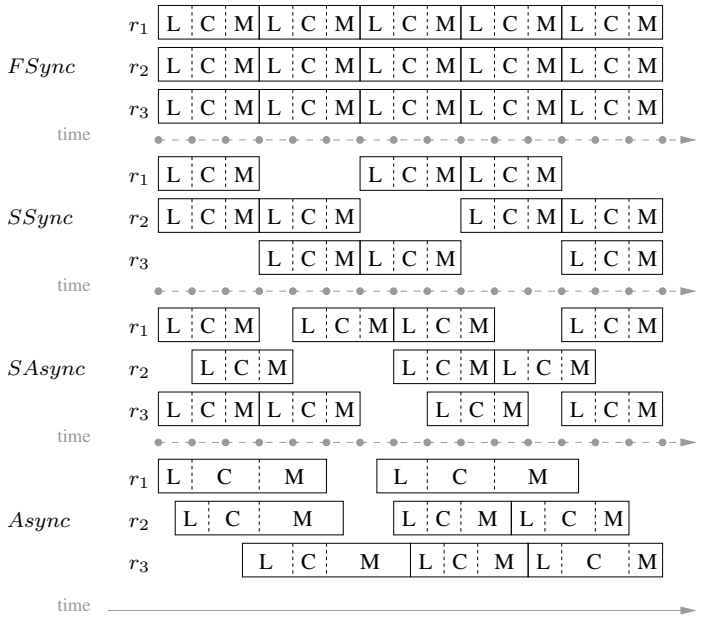


Figure 3.1: The execution model of computational cycles for each of FSYNC, SSYNC, SASYNC, and ASYNC robots. The inactivity of robots is implicitly represented by empty time periods.

of an algorithm. This also means that the actions of the robots do not depend on past cycles. In this sense, the system is robust to memory failures. A robot perceives any configuration as if it were an initial configuration. A robot must be able to compute a move from it so that the system does not end in an inconsistent, unpredicted state from which the system cannot recover.

3.1.1 Varying the components of the system

We just listed the minimal capabilities of the robots in the $\mathcal{OBLOT}$ model, however, each of the assumptions can be weakened depending on the scenario we want to model and on the task of robots. Given a task, the goal is to determine what are the minimum capabilities needed by the robots to perform it. Sometimes we need to relax some of the assumptions presented in Section 3.1 to fulfill a task like for example allowing unlimited visibility if the task is not achievable only with limited visibility.

Time scheduler. Robots' behavior is defined by the repetition of the Look-Compute-Move cycle. Assumptions made on the duration of the robots' cycle and the time at which robots are activated have a great impact on the capabilities of the robots. A time scheduler decides at which time robots are activated following one of these four different models:

- *Fully-synchronous* (FSYNC): the activation of the robots is logically divided into global rounds; all the robots are activated in every round, start the cycle simultaneously and execute it synchronously. It corresponds to the hypothesis that robots share a common clock.

- *Semi-synchronous* (SSYNC): it differs from the FSYNC scheduler for the fact that in each round, not all the robots are activated whiles others are in Wait state. The choice of which robots are activated in a given round is assumed to be made by the time scheduler.

- *Semi-Asynchronous* (SASYNC): Robots are activated independently. Like in FSYNC or SSYNC, the duration of each phase is assumed to be always the same. Differently from FSYNC or SSYNC, two activated robots can be in different phases even though phases are synchronized.

- *Asynchronous* (ASYNC): The robots are activated independently from others and each phase of the cycle is finite but can have an unpredictable duration. In other words, robots do not have a common notion of time. Moreover, since the Look phase is instantaneous a robot cannot perceive if other robots are moving or not. As a result, computations can be made based on totally obsolete observations, taken arbitrarily far in the past. As a robot starts moving, the configuration of robots might be different from the one perceived during the look phase.

In schedulers different from FSYNC, it is necessary to guarantee that the time scheduler is **fair**: for every robot r and time t, there exists a time $t' \geq t$ at which r is activated; that is, every robot is activated infinitely often.

Clearly, the four synchronization schedulers induce the following hierarchy (see, e.g. [43, 53, 60]): FSYNC robots are more powerful (i.e., they can solve more tasks) than SSYNC robots, that in turn are more powerful than SASYNC robots, that in turn are more powerful than ASYNC robots. This simply follows by observing that the adversary can control more parameters in ASYNC than in SASYNC, and it controls more parameters in SASYNC than in SSYNC and FSYNC. In other words, protocols designed for ASYNC robots also work for SASYNC, SSYNC and FSYNC robots. On the contrary, any impossibility result stated for FSYNC robots also holds for SSYNC, SASYNC and ASYNC robots.

Orientation. In general, robots are assumed to be disoriented: each of them has its own LCS and its unit of measure. It is possible to customize the system by assuming that all robots agree on the direction and orientation of k axes ($1 \leq k \leq d$). If robots have *chirality*, they agree on a cyclic orientation (e.g., clockwise) of the plane.

Movements. An external *mobility scheduler* controls the movement of a mobile robot. The scheduler determines how quickly the robot moves toward its destination

point, and it may even stop its movement before it arrives. Two variants can be defined: *rigid* (or unlimited) mobility, where all robots always reach their destinations when performing `Move`; *non-rigid*, where the distance traveled within a move is neither infinite nor infinitesimally small. More precisely, even if the mobility scheduler can stop a robot before it reaches its destination, there exists an unknown constant $\delta > 0$ such that if the destination point is closer than δ, the robot will reach it, otherwise the robot will be closer to it of at least δ.

Extent. Robots are viewed as points, in the standard model, i.e., they are dimensionless. This property can be changed by assuming robots with a physical dimension, that is, entities with an extent. These robots are called *solid* (or *fat* as in [13, 29, 48]) and are viewed as opaque circular disks of fixed diameter (hence they are assumed to have a common unit distance).

Memory, Appearance and Communication. Robots can be endowed with a persistent and externally visible state variable, called *visible light*, that can assume values from a finite set of colors. The light can be set in each cycle by the robot at the end of its `Compute` operation. It is externally visible because its color at time t is visible to all robots in its visibility radius that perform a `Look` operation at that time. It is persistent because the variable is not automatically reset at the end of a cycle. The color a robot sees is used as input during the computation. Luminous robots can be seen as robots having persistent information used to remember and communicate. Moreover, color affects the appearance of a robot, only robots having the same color are identical.

Adversary. The mobility scheduler as well as the time scheduler are both managed by an ideal *adversary*. Such schedulers are completely out of the control of the robots. This does not imply that the environment is centralized, but rather that any event is possible. However, the occurrence of an event is not just thought as a random process, since otherwise one may infer some properties with high probability. The adversarial technique, instead, is a way to keep in mind the worst-case scenario.

Algorithm. Regardless of the adversary, the activations of the robots determine specific ordered time instants. Let $R(t)$ be the configuration observed by some robots at time t during their `Look` phase, and let $\{t_i : i = 0, 1, \ldots\}$, with $t_i < t_{i+1}$, be the set of all time instances at which at least one robot takes the snapshot $R(t_i)$. Since the information relevant for the computing phase of each robot is the order in which the different snapshots occur and not the exact time in which each snapshot is taken, then, without loss of generality we can assume $t_i = i$ for all $i = 0, 1, \ldots$. It follows that an *execution* of an algorithm $\mathcal{A}$ from an initial configuration R is a sequence of configurations $\mathbb{E} : R(0), R(1), \ldots$, where $R(0) = R$ and $R(t + 1)$ is obtained from $R(t)$ by moving some robots according to the result of the `Compute` phase as implemented by $\mathcal{A}$. Moreover, given an algorithm $\mathcal{A}$, there exist many different executions from $R(0)$ depending on the activation and the movement of the robots, controlled by the adversary.

Symmetric configurations and Symmetricity

Let $d()$ be the function computing the Euclidean distance between points in the plane, and let φ any map from points to points in the plane: φ is called an *isometry* if $d(\varphi(a), \varphi(b)) = d(a, b)$ for any $a, b \in \mathbb{R}^2$. Examples of isometries in the plane are rotations and reflections. An isometry φ is a rotation if there exists a unique point x such that $\varphi(x) = x$ (and x is called *center of rotation*); it is a reflection if there exists a line ℓ such that $\varphi(x) = x$ for each point $x \in \ell$ (and ℓ is called *axis of symmetry*).

An isometry φ may induce an isometry for a configuration of robots: if a robot r is mapped into a robot r' then φ maps the location of r into the location of r'. This implies that a multiplicity of $k \geq 1$ robots is always mapped into a multiplicity with the same value. The isometries for configurations are the identity, rotations, reflections and their compositions. If R admits only the identity, then R is said **asymmetric**, otherwise it is said **symmetric** (i.e., R admits rotations or reflections). Any configuration that admits a multiplicity is symmetric.

In a symmetric configuration R, consider any subset of pairwise symmetric robots: such robots are in fact **equivalent** as their symmetry gives no rise to any means for an algorithm to distinguish among them. This is better explained in the following remark.

Remark 1. *Let R be a symmetric configuration. It can be observed that there exists a set of local coordinate systems for robots located in R that is symmetric with respect to the center of R or to an axis of symmetry. Hence, if we consider any subset R' of pairwise equivalent robots in R, being the robots identical and homogeneous, then any move planned by any algorithm $\mathcal{A}$ for any robot in R' will be applied to* all *the robots in R'. It follows that the adversary may force all the robots in R' to perform symmetric movements and hence it results to be impossible to break the symmetry among the robots in R'.*

Consider now a symmetric configuration R composed of asynchronous robots. According to Remark 1, no algorithm can avoid that two (or more) equivalent robots in R start the computational cycle simultaneously. In such a case, there might occur a so called **pending move**, that is, one of the two robots performs its entire computational cycle while the other has not started or not yet finished its `Move` phase. Formally, a robot r aims to perform a pending move in a configuration $R(t)$, if at time t robot r is active, has taken a snapshot $R(t') \neq R(t)$ with $t' < t$, and is planning to move or is moving with a non-*nil* trajectory. Clearly, any other robot r' is not aware whether r aims to perform a pending move, that is it cannot deduce such information from the snapshot acquired in the `Look` phase. This fact greatly increases the difficulty to devise algorithms for ASYNC robots in symmetric configurations. All these difficulties are overcomed if an algorithm is able to produce always **stationary configurations**: a configuration $R(t)$ is called *stationary* if there are

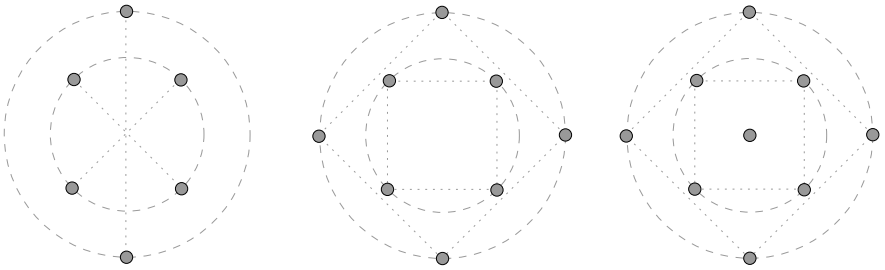


Figure 3.2: Examples of symmetricity of a set of points P. In (a), P is partitioned in regular 2-gons and $\rho(P) = 2$; in (b), the maximum m for which there is a partition of P into regular polygons is four and $\rho(P) = 4$; in (c), P can only be partitioned in 1-gons, thus $\rho(P) = 1$.

no pending robots in $R(t)$. When an algorithm moves a robot at a time, then the obtained configuration is stationary by definition.

Symmetricity. Let P be a set of points in the Euclidean plane, $\overline{C}(P)$ be the smallest circle enclosing all the points in P, and $c(P)$ be the center of $\overline{C}(P)$. We consider a decomposition of P into regular m-gons with a common center, where one point forms a regular 1-gon with an arbitrary center, and two points form a regular 2-gon with the center being the midpoint. Then, we consider the maximum value of m. In [131], this maximum value m is called the **symmetricity** of P and it is denoted by $\rho(P)$. Accordingly, given a set P with n points, the concept of symmetricity induces a unique partition $\mathcal{P} = \{P_1, P_2, \ldots, P_k\}$ of P into k subsets, where each subset P_i forms a ρ-gon and $k = n/\rho(P)$. When $\rho(P) > 1$ the symmetricity represents the rotational symmetry of P, and the center of the symmetry coincides with $c(P)$. See Figure 3.2 for some examples about $\rho(P)$. In Figure 3.2.(a), P can be partitioned either into six 1-gons or into three 2-gons, therefore the maximum value of m in which P can be partitioned equals two, therefore $m = 2$ and the symmetricity $\rho(P) = 2$. In Figure 3.2.(b), P can be partitioned into $8/m$ regular m-gons for $m = 1, 2, 4$ and $\rho(P) = 4$. In Figure 3.2.(c), the only regular partition of P is into nine 1-gons, because the central node does not belong to any m-gons with $m > 1$, therefore $\rho(P) = 1$. All the above concepts and notation can be directly applied to any configuration by considering the points in the plane occupied by the robots in R. Hence, we can use notation as $c(R)$ and $\overline{C}(R)$ and the concept of symmetricity of R as well (i.e., $\rho(R)$ corresponds to the symmetricity of the points in the plane occupied by all robots in R).

A symmetric configuration R with $\rho(R) = 1$ (cf. Figure 3.2.c) can be modified into an asymmetric one by moving the central robot away from $c(R)$.

Robots moving on graphs

Configurations. The topology where robots are placed on is represented by a simple, undirected, and connected graph $G = (V, E)$, with vertex set V and edge set E. A function $\lambda : V \to \mathbb{N}$ represents the number of robots on each vertex of G, and we call $C = (G, \lambda)$ a *configuration* whenever $\sum_{v \in V} \lambda(v)$ is bounded and greater than zero. A vertex $v \in V$ such that $\lambda(v) > 0$ is said *occupied*, *unoccupied* otherwise. A *multiplicity* occurs in any vertex $v \in V$ such that $\lambda(v) > 1$.

Movements. At most one edge can be traversed during the `Move` phase. The robot either stays on the vertex where it currently resides (performs a *nil* movement) or moves to a vertex among those at one hop distance. Movements on a graph are always considered instantaneous. From that follows that robots are always on vertices and never on edges during Look phases. Hence, robots cannot be seen while moving, but only at the moment they may start moving or when they arrive. The rationale behind this assumption is that the graph may model a communication network, whereas robots model software agents.

Symmetric configurations. Two undirected graphs $G = (V, E)$ and $G' = (V', E')$ are *isomorphic* if there is a bijection φ from V to V' such that $\{u, v\} \in E$ if and only if $\{\varphi(u), \varphi(v)\} \in E'$. An *automorphism* on a graph G is an isomorphism from G to itself, that is a permutation of the vertices of G that maps edges to edges and non-edges to non-edges. The set of all automorphisms of G, under the composition operation, forms a group called *automorphism group* of G and denoted by $\mathrm{Aut}(G)$. If $|\mathrm{Aut}(G)| = 1$, that is G admits only the identity automorphism, then G is said *asymmetric*, otherwise it is said *symmetric*. Two distinct vertices $u, v \in V$ are *equivalent* if there exists an automorphism $\varphi \in \mathrm{Aut}(G)$ such that $\varphi(u) = v$.

The concept of graph automorphism can be extended to configurations in a natural way: (1) two configurations $C = (G, \lambda)$ and $C' = (G', \lambda')$ are isomorphic if G and G' are isomorphic via an isomorphism φ and $\lambda(v) = \lambda'(\varphi(v))$ for each vertex v in G; (2) an automorphism of a configuration $C = (G, \lambda)$ is an isomorphism from C to itself, and (3) the set of all automorphisms of C forms a group under the composition operation that we call automorphism group of C and denote as $\mathrm{Aut}(C)$. Moreover, if $|\mathrm{Aut}(C)| = 1$ we say that C is *asymmetric*, otherwise it is *symmetric*. Two distinct robots r and r' in a configuration (G, λ) are *equivalent* if there exists $\varphi \in \mathrm{Aut}(C)$ that makes equivalent the vertices in which they reside. Note that $\lambda(u) = \lambda(v)$ whenever u and v are equivalent. Moreover, if u and v are equivalent, a robot r cannot distinguish its position at vertex u from robot r' located at vertex $v = \varphi(u)$. As a consequence, no algorithm can distinguish between two equivalent robots.

3.2 The methodology adopted for algorithm design

The coordination of distributed agents is a challenging algorithmic task because the capabilities of the robots are reduced to a minimum. For example, while attempting to guide a set of robots to form a geometric shape, the lack of agreement on a global coordinating system makes it hard for robots to agree on the target positions. Moreover, in the *Async* setting, some robots move while others perform another phase. As a result, robots may fail to recognize the stage reached by the algorithm, and this incorrect detection may make it impossible to finalize the algorithm. Given a configuration of robots, a distributed algorithm moves the robots to reach a final configuration having some properties. The only information robots have access to is the view of the position of the other robots. As robots move and their position change, the system can potentially assume an exponential number of different configurations. Robots must be able to recognize the stage of the algorithm just watching the configuration of robots.

The distributed algorithms presented in this thesis are designed following a hierarchical decomposition introduced by Cicerone et al. in [44]. Each problem is divided into sub-problems easy enough to be tackled by a group of $\mathcal{OBLOT}$ robots. The decomposition approach allows to prove the correctness of each phase. Given a problem Π we characterize when it is potentially solvable, and for all these input configurations we design a resolution algorithm A. The strategy is to divide Π into independent sub-problems called tasks T_i. Each task will bring the robots into an intermediate configuration. The topological features of a configuration of robots are described as a predicate that consists of variables combined in Boolean logic. A predicate P_i is defined for each of the tasks T_i and it is a precondition to the execution of the corresponding task. Predicates must be mutually exclusive. Each robot evaluates the predicates during the Compute phase until it finds one that is true. When a robot finds that a predicate P_i is true, it knows that the corresponding task must be performed. Each task has also an associated move m_i. A move identifies a subset of robots S to move and describes their trajectory. In the move phase the robots in S perform move m_i. The design of the preconditions is difficult because it must guarantee two main properties: it has to identify unique topological features for each task, and these don't have to vary during the movements of the robots and until a task is completed, and we must be sure that a robot is able to compute each of the variables which compose a predicate given the information acquired during the look phase. Suppose a robot is stopped by the adversary during the execution of its move, when the robot starts a new LCM cycle, it evaluates again the preconditions, and if it has the same outcome, it will perform the same move. Predicates are defined by:

- *basic variables* that capture metric/topological/numerical/ordinal aspects of the input configuration which are relevant for the used strategy and that can be evaluated by each robot on the basis of its view;

- *composed variables* that express the pre-conditions of each task T_i.

A formal approach requires that predicates must guarantee some properties:

- Prop_1*: each* P_i *must be computable on the configuration perceived in each* `Look` *phase;*
- Prop_2*:* $P_i \wedge P_j = \texttt{false}$*, for each* $i \neq j$*;*
- Prop_3*: for each possible perceived configuration there must exists a predicate* P_i *evaluated as true.*

Each task can be accomplished only when its precondition is fulfilled. For the sake of simplicity, denote as $\texttt{pre}_i$ the precondition defined for T_i for each $2 \leq i \leq n$ with n the total number of tasks and define $\texttt{pre}_1 = \mathit{true}$. Then, P_i can be formally defined as follows:

$$P_i = \texttt{pre}_i \wedge \neg \bigvee_{j>i} \texttt{pre}_j. \tag{3.1}$$

Equation 3.1 assumes the ordering of the tasks. The first precondition evaluated by the robots is the last one $\texttt{pre}_n$, if it's false, then $\texttt{pre}_{n-1} \wedge \neg\texttt{pre}_n$ is evaluated. If all the preconditions up to $\texttt{pre}_2$ are evaluated false, then task T_1 is performed on an input configuration. Given a configuration of robots at time t, if each of the robots performs the compute phase they must be able to recognize the same task to perform, in other words they must agree on the task. When a task T_i completes, we say the algorithm transitions from a task T_i to task T_j if there exists an input configuration and an execution of the algorithm that generates such a transition. The set of all the transitions of A are defined in the transition graph in which V is the set of tasks T_i and E is the set of directed edges (T_i, T_j) if exists a transition from T_i to task T_j. Tasks, preconditions, moves and transitions will be summarized in a table.

The methodology proposed in [44] allows us to prove formally the correctness of an algorithm by proving that all the following properties hold:

H_1**:** *The algorithm never generates unsolvable configurations.*

H_2**:** *The movement of each robot is collision-free.*

H_3**:** *For each task* T_i*, the transitions from* T_i *to any other task are "exactly" those declared in the transition graph.*

H_4**:** *Each transition in a graph occurs after a finite number of cycles. This means that the generated configurations can remain in the same task only for a finite number of cycles.*

All these properties must be proved for each transition or move.

3.3 Arbitrary patterns on tessellation graphs

We consider the *Arbitrary Pattern Formation* (*APF*) problem. Given a set R of robots, each one located at a different vertex of an infinite regular tessellation graph, we design a distributed algorithm that guides the robots to form any specific but arbitrary geometric pattern given in input.

So far, under the $\mathcal{OBLOT}$ model, the *APF* problem has been investigated only on the regular square grids. Grids are a natural discretization of the Euclidean plane. Other notable regular tessellations are triangular and hexagonal grids. In particular, the triangular grid is the most general in terms of possible symmetries and trajectories. We present an algorithm for *APF* when the initial configuration of robots is **asymmetric** and the considered topology is any regular tessellation graph. The algorithm is first described in detail with respect to the triangular grid, then we revisit the algorithm with respect to both the square and the hexagonal grids, pointing out any possible deviations required with respect to the specific graph classes. Robots are modeled according to the $\mathcal{OBLOT}$ model (e.g., [73]), **asynchronous** and endowed with **global strong multiplicity detection**. As robots move on graphs, in the `Move` phase at most one edge can be traversed. Initially, robots are inactive, but once the execution of an algorithm $\mathcal{A}$ starts - unless differently specified - there is no instruction to stop it, i.e., to prevent robots to enter their LCM cycles. Then, the *termination* property for $\mathcal{A}$ can be stated as follows: once robots have reached the required goal through $\mathcal{A}$, from there on robots can perform only the *nil* movement. We assume that cycles are performed according to the weakest Asynchronous scheduler (ASYNC) (cf. [22, 36, 37, 40, 54, 72, 79]).

The activation of the robots determines specific ordered time instants. Let $C(t)$ be the configuration observed by some robots at time t during their `Look` phase, and let $\{t_i : i = 0, 1, \ldots\}$, with $t_i < t_{i+1}$, be the set of all time instances at which at least one robot takes the snapshot $C(t_i)$. Since the information relevant for the computing phase of each robot is the order in which the different snapshots occur and not the exact time in which each snapshot is taken, then without loss of generality we can assume $t_i = i$ for all $i = 0, 1, \ldots$. Then, an *execution* of an algorithm $\mathcal{A}$ from an initial configuration C is a sequence of configurations $\mathbb{E} : C(0), C(1), \ldots$, where $C(0) = C$ and $C(t + 1)$ is obtained from $C(t)$ by moving some robot according to the result of the `Compute` phase as implemented by $\mathcal{A}$. This definition of execution works also for the other schedulers. Moreover, given an algorithm $\mathcal{A}$, in ASYNC (but also in SASYNC and SSYNC) there exists more than one execution from $C(0)$ depending on the activation of the robots (which depends on the adversary).

Previous work

A restricted version of APF has been first solved in [65] for robots moving on the Euclidean plane. The algorithm requires at least $n \geq 4$ asynchronous robots equipped with chirality and, the possible patterns cannot have multiplicities. In particular, the configurations from which the proposed algorithm could output any pattern are the so-called *leader configurations*. These are configurations of robots (including some symmetric ones) in which it is possible to elect a leader. In [26, 139], authors remove these restrictions using randomization techniques. APF is solved through a deterministic algorithm for robots without chirality allowing patterns with multiplicities in [37]. Further investigations of APF in the Euclidean plane referring to slightly different models can be found in [25, 71]. When multiplicities are allowed in output patterns, the degenerate case of point formation, Gathering, is included in APF. The gathering problem has been fully characterized in [45]. A hybrid environment where robots move in the Euclidean plane but can accomplish the gathering task only at predetermined points has been studied in [36]. APF has been recently addressed in [24] for robots moving on graphs, and in particular, on an infinite square grid. The initial configurations are asymmetric and the output patterns cannot have multiplicities, therefore gathering is not included. Gathering on infinite or finite square grids has been fully characterized in [52, 63], also considering the minimization of the overall traveled distances. All the results for the gathering problem achieved so far, for general graphs or specific graph topologies are in [39, 42].

Outline

The next section defines the problem formally and introduces the notation used in the designed algorithm, called $\mathcal{A}_{form}$. Section 3.3.3 provides a high-level description of $\mathcal{A}_{form}$. Section 3.3.4 formalizes the algorithm and provides the correctness. Since all the details are given for the triangular grid, in Section 3.3.5, we revisit the algorithm for both the square and the hexagonal grids. Section 3.3.6 highlights some final remarks.

3.3.1 Problem definition and basic notation

Configurations on tessellation graphs.

In this work, we consider G as an infinite graph generated by a *plane tessellation*. A tessellation is a tiling of a plane with polygons without overlapping. A *regular tessellation* is a tessellation that is formed by just one kind of regular polygon of side length 1 and in which the corners of polygons are identically arranged. According to [83], there are only three regular tessellations, and they are generated by squares, equilateral triangles or regular hexagons (see Figure 3.3). An infinite lattice of a regular tessellation is a lattice formed by taking the vertices of the regular polygons

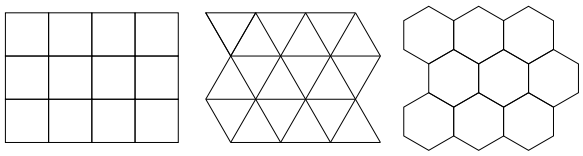


Figure 3.3: Part of regular plane tessellations.

in the tessellation as the points of the lattice. A graph G is induced by the point set S if the vertices of G are the points in S and its edges connect vertices that are distance 1 apart. A *tessellation graph* of a regular tessellation is the infinite graph embedded into the Euclidean plane induced by the infinite lattice formed by that tessellation [89]. We denote by G_S (G_T and G_H, respectively) the tessellation graphs induced by the regular tessellations generated by squares (equilateral triangles and regular hexagons, respectively). A function $\lambda : V \to \mathbb{N}$ represents the number of robots on each vertex of G. We call $C = (G, \lambda)$ a *configuration* where $G \in \{G_S, G_T, G_H\}$ and whenever $\sum_{v \in V} \lambda(v)$ is bounded and greater than zero. A vertex $v \in V$ such that $\lambda(v) > 0$ is said *occupied*, *unoccupied* otherwise and A *multiplicity* occurs in any vertex $v \in V$ such that $\lambda(v) > 1$.

Definition 1. *Given a graph $G \in \{G_S, G_T, G_H\}$, any line parallel to any edge of G is called* canonical direction*. The smallest angle formed by the available canonical directions is called the* canonical angle*.*

According to Definition 1, in G_S there are just two canonical directions and the canonical angle is of 90°. In both G_T and G_H there are three canonical directions and the canonical angle is of 60°.

Concerning the configurations addressed in this work, any $C = (G, \lambda)$, with $G \in \{G_S, G_T, G_H\}$, admits only two types of automorphisms: *reflections*, defined by an axis of reflection that acts as a mirror; *rotations*, defined by a center and an angle of rotation. The axes of reflection are of two types: the ones of the considered regular polygons and those coincident with any side of the regular polygons. The centers of possible rotations are only on specific points of the regular polygons: on the center, on one vertex, or on the middle point of a side. The rotation angle is specific of each given tessellation graph.

3.3.2 Problem formalization

A configuration $C = (G, \lambda)$, with $G = (V, E)$, is *initial* if both the following conditions hold: (1) each robot is idle and placed on a different vertex, that is $\lambda(v) \leq 1$ for each $v \in V$; (2) C is asymmetric. The set containing all the initial configurations is denoted by $\mathcal{I}$.

The goal of the *APF* problem is to design a distributed algorithm $\mathcal{A}$ that guides the robots to form a fixed arbitrary pattern F starting from any configuration $C = (G, \lambda)$ such that $G \in \{G_S, G_T, G_H\}$ and $C \in \mathcal{I}$. The pattern F is a multi set of vertices, given in any coordinate system, indicating the corresponding target vertices in the tessellation graph G. It constitutes the input for all robots. Due to the absence of a common global coordinate system, the robots decide that the pattern is formed when the current configuration becomes "similar" to F with respect to translations, rotations, and reflections. The problem can be formalized as follows: an algorithm $\mathcal{A}$ solves the *APF* problem for an initial configuration C if, for each possible execution $\mathbb{E} : C = C(0), C(1), \ldots$ of $\mathcal{A}$, there exists a finite time instant $t^* > 0$ such that $C(t^*)$ is similar to F and no robot moves after t^*, i.e., $C(t) = C(t^*)$ holds for all $t \geq t^*$.

Notation

Here we introduce some concepts and notation used in the algorithm. Given a configuration $C = (G, \lambda)$, we use $R = \{r_1, r_2, \ldots, r_n\}$ to denote the set containing all the n robots located on G (we recall that robots are anonymous and such a notation is used only for the sake of presentation). The distance $d(u, v)$ between two vertices $u, v \in V$ is the number of edges of a shortest path connecting u to v. We extend the notion of distance to robots: $d(r_i, r_j)$ denotes the distance between the two vertices in which the robots reside. Symbol $D(r)$ is used to denote the *sum of distances* of $r \in R$ from any other robot, that is $D(r) = \sum_{r_i \in R} d(r, r_i)$.

Given a set of points P in the plane, $mbr(P)$ represents the *minimum bounding rectangle* of P, that is the rectangle enclosing all the points in P and fulfilling the following properties: (1) its sides are all parallel to the Cartesian axes, and (2) each pair of its parallel axes are as close as possible. According to the definition, we get that $mbr(P)$ is unique.

This definition of a minimum bounding rectangle can be easily extended to a set of robots R placed on the tessellation graph G_S where the canonical directions are just two, and they can naturally play the role of the Cartesian axes. Unfortunately, it does not work when R is placed on tessellation graphs such as G_T or G_H. To generalize it, we move to the concept of *bounding parallelogram* $bp(R)$, defined as any parallelogram enclosing all robots, with sides parallel to two of the three available canonical directions, and with each pair of parallel sides as close as possible. Since G_T or G_H admit three canonical directions, it can be observed that the bounding parallelogram of R is not unique. In fact, there are three possible bounding

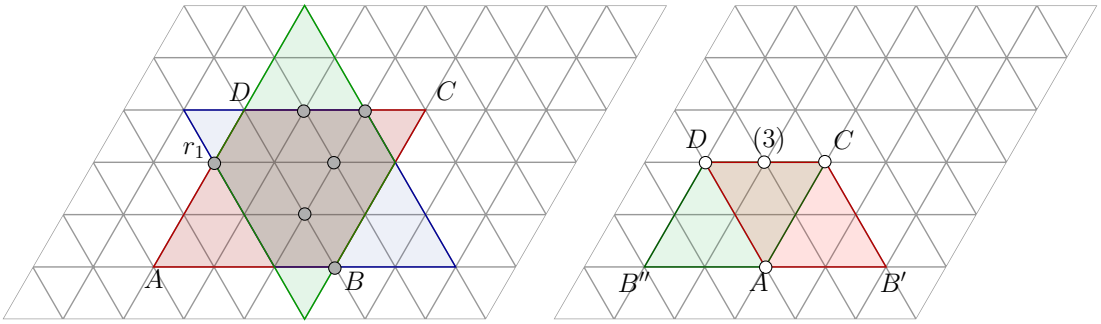


Figure 3.4: *Filled circles represent robots, empty circles represent elements of a pattern F. (left)* An initial configuration C with $n = 6$ robots. It shows that $bp(R)$ is not unique in G_T. The red parallelogram generates the *LSS*. The leading corner is A and the leading direction is AB. The unique *LSS* is $\ell = (0,0,0,1,\ 0,0,1,0,\ 1,0,1,0,\ 0,1,1,0)$. Robot r_1 has maximum sum of distances, with value $D(r_1) = 13$. *(right)* A possible pattern F to be formed. The number close to a vertex refers to a multiplicity. Since F is symmetric, there are two (equivalent) $mbp(F)$.

parallelograms (e.g., see Figure 3.4).

Given any $bp(R)$, we denote by $H(bp(R))$ and $\texttt{Mf}(bp(R))$, with $H(bp(R)) \leq \texttt{Mf}(bp(R))$, the width and height of $bp(R)$, respectively. Similarly, $H(bp(F))$ and $\texttt{Mf}(bp(F))$ are used to denote the same values with respect to $bp(F)$.

Let $bp(R)$ be any bounding parallelogram of R. We associate a sequence of integers to each canonical corner of $bp(R)$ (e.g., corners A and C in Figure 3.4). The sequence associated with a canonical corner A is defined as follows. Scan the finite grid enclosed by $bp(R)$ from A along $h(bp(R))$ (say, from A to B) and sequentially all grid lines parallel to AB in the same direction. For each grid vertex v, put $\lambda(v)$ in the sequence. Denote the obtained sequence as $s(AB)$. Being $h(bp(R)) = w(bp(R))$ in the example, from A it is also possible to obtain the sequence $s(AD)$, and hence four sequences can be defined in total, two for the corner A and two for the corner C. If any two of these sequences are equal, then it implies that the configuration admits a (reflectional or rotational) symmetry. We denote by *LSS* the lexicographically smallest sequence. It is unique by definition.

The canonical corner from which a *LSS* starts is called the *leading corner*; the canonical direction from the leading corner used to create the *LSS* is called the *leading direction*. The *LSS* of a given $bp(R)$ is denoted by $\ell(bp(R))$, or simply by ℓ when $bp(R)$ can be inferred by the context.

Definition 2. *Let $C = (G, \lambda)$ be a configuration with $G \in \{G_S, G_T, G_H\}$ and set of robots R. A* minimum bounding parallelogram *$mbp(R)$ is defined as any bounding parallelogram $bp(R)$ with $h(bp(R))$ minimum, and with minimum LSS in case of ties.*

It can be easily observed that any asymmetric configuration admits exactly one $mbp(R)$ whereas symmetric configurations admit multiple $mbp(R)$'s. However, the LSS's associated to such $mbp(R)$'s are all the same.

3.3.3 Algorithm description

In this section, we provide a high-level description of the algorithm $\mathcal{A}_{form}$ to solve APF for any initial configuration $C = (G_T, \lambda)$ of n ASYNC robots endowed with the global strong multiplicity detection and with all the minimal capabilities recalled in Section 3.1. We assume $n \geq 3$, since for $n = 1$ the APF problem is trivial and for $n = 2$ we get that C is symmetric and hence not initial. Concerning the pattern F, it might contain multiplicities.

The strategy

Following this approach explained in Section 3.2, APF is initially divided into four sub-problems denoted as Reference System (RS), Partial Pattern Formation (PPF), Finalization (Fin), and Termination ($Term$). Some of these sub-problems are further refined until the corresponding tasks can be suitably formalized according to the assumed capabilities of the robots. This leads to the following decomposition:

- *Reference System (RS = How to embed F on G_T).* This sub-problem addresses one of the main difficulties arising from the general pattern formation problem: the lack of a unique embedding of F on G_T that allows each robot to uniquely identify its target (the final destination in the pattern). In particular, RS moves or matches a minimal number of robots into specific positions so that they form a common reference system for any other robot. These robots are called *guards*. The reference system implies a unique mapping from robots to targets, keeping the mapping during the movements of robots. In our strategy RS is further divided into three sub-problems denoted as RS_{1a}, RS_{1b}, and RS_2. These sub-problems are associated with three tasks named T_1, T_2, and T_3, respectively. The first two are devoted to placing the first guard, denoted as r_1, whereas the third fixes the position of a second guard denoted as r_n. Once the two guards reach their positions, the reference system is given by two lines passing through the vertices occupied by the guards and forming a canonical angle between them. When the reference system is created, all the robots, except the guards, result to be located in a specific quadrant called Q^-.

- *Partial Pattern Formation (PPF = How to form part of F).* This sub-problem is associated with task T_4, and it activates only once RS is solved. It forms a pattern similar to F by using robots in $R'' = R \setminus \{r_1, r_n\}$. Thanks to the common reference system, all robots agree on the embedding of F on a

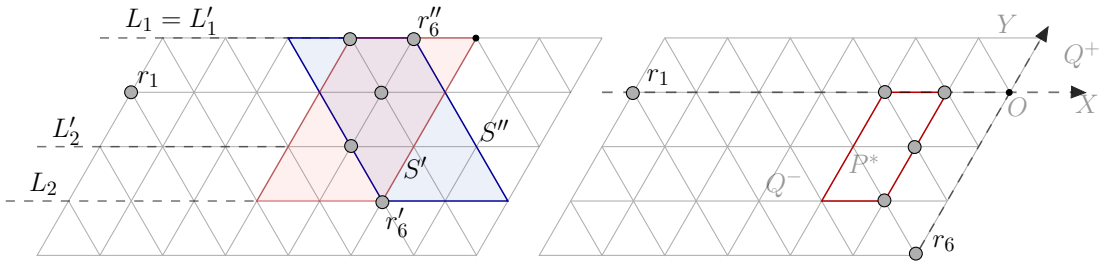


Figure 3.5: *(left)* Visualization of task T_1 concerning the initial movement of r_1 (cf. configuration C in Figure 3.4). *(right)* Visualization of task T_2 concerning the final destination of r_1. Once r_1 stops, all the items necessary to define the reference system can be settled (cf., Remark 2).

quadrant denoted as Q^+ that is different from Q^-. During the task, all the $n-2$ robots in R'' move from Q^- to Q^+. Robots move one at a time so that there are no collisions.

- *Finalization (Fin = How to move r_1 and r_n so that F is formed).* The finalization task activates when the guards are the only robots not well positioned according to F. While the guards r_1 and r_n move, the common reference system is lost. However, we can guarantee that robots always detect that they are in the finalization phase. The two robots reach their targets and complete the pattern F. *Fin* is divided into three tasks: T_5 is for the movement of r_n, whereas T_6 and T_7 relates to the movement of r_1.

- *Termination (Term).* Robots need to recognize that the pattern is complete and that further movements are not required. Only *nil* movements are allowed and it is impossible to switch to any other task. Task T_8 relates to the termination phase.

In the rest of the section, we detail each task.

Task T_1. It selects a robot denoted as r_1 (the first guard) such that $D(r_1)$ is maximum (cf., Figure 3.4). In case of ties, r_1 has the minimum position in $\ell(mbp(R))$ – recall that the input configuration is asymmetric and hence $mbp(R)$ is unique. Let $R' = R \setminus \{r_1\}$, during this task r_1 moves through any shortest path toward the closest vertex that satisfies the following Boolean variable:

- `g1` = *there exists a unique line parallel to a canonical direction passing through r_1 and each $bp(R')$.*

Note that, when `g1` holds we identify the unique line passing through r_1 and each $bp(R')$ as the *line induced* by `g1`.

Task T_2. In this task, we assume variable `g1` true, holding at the end of the task T_1 – this is a *precondition* to perform T_2. The goals of this task are: (1) to move r_1 so that its position defines the X-axis, (2) to identify a second guard r_n.

When the task starts, r_1 is the robot r such that $D(r)$ is maximum whereas the second guard r_n is identified as follows:

- Let L be the line induced by `g1`. It can be observed that there are exactly two distinct $bp(R')$'s with sides parallel to L. Let L_1 and L_2 be the two lines parallel to L shared by the two $bp(R')$'s (cf., Figure 3.5). Denote the two $bp(R')$'s as P' and P'', and denote as S' (S'', respectively) the side of P' (P'', respectively) which lies neither on L_1 nor on L_2 and is further from r_1. In particular, P' (P'', respectively) is the parallelogram having the canonical angle formed by the intersection of S' (S'', respectively) and L_1 (L_2, respectively) – the red parallelogram in Figure 3.5. Denote as r'_n (r''_n, respectively) the robot on S' (S'', respectively) closest to L_1 (L_2, respectively). The second guard useful to define the reference system is selected between r'_n and r''_n.

  Robot r_1 considers the line L'_1 (L'_2, respectively) defined as L_1 (L_2, respectively) but referred to $R' \setminus \{r'_n\}$ ($R' \setminus \{r''_n\}$, respectively) instead of R'. Then r_1 selects the closest line between L'_1 and L'_2 (it arbitrarily selects one of the two in case of ties). Without loss of generality, assume that r_1 selects L'_1. According to this choice, r_1 promotes r'_n to be r_n, that is the second guard (symmetrically, if r_1 selects L_2, then r''_n is promoted).

After computing the second guard, the robots have enough information to identify a common reference system, as stated in the following remark.

Remark 2. After computing the second guard, robots have sufficient information to compute a common reference system. In fact, the line between L'_1 and L'_2 selected by r_1 defines the X-axis, and this axis must be intended as directed from r_1 to all the other robots; all vertices in the half-plane containing robots in R' are considered with negative Y-coordinates. The second guard r_n is induced by the line between L'_1 and L'_2 selected by r_1 (as described above). The line passing through r_n, intersecting the X-axis, and forming a canonical angle in the first quadrant defines the Y-axis. Finally, the intersection between the two axes defines the origin of the system denoted as O. In this reference system, the first quadrant is denoted as Q^+, while the third quadrant is denoted as Q^-.

The target of r_1 (r_n, respectively) is on the X-axis (Y-axis, respectively) at a distance from the origin, ensuring that the configuration stays asymmetric during the subsequent *PPF* task. Robots compute the distance as follows:

- Let R^* be the (possibly empty) subset of robots of R'' lying in Q^-, P^* be the parallelogram $bp(R^*)$ having the directions parallel to the X- and Y-axes, and

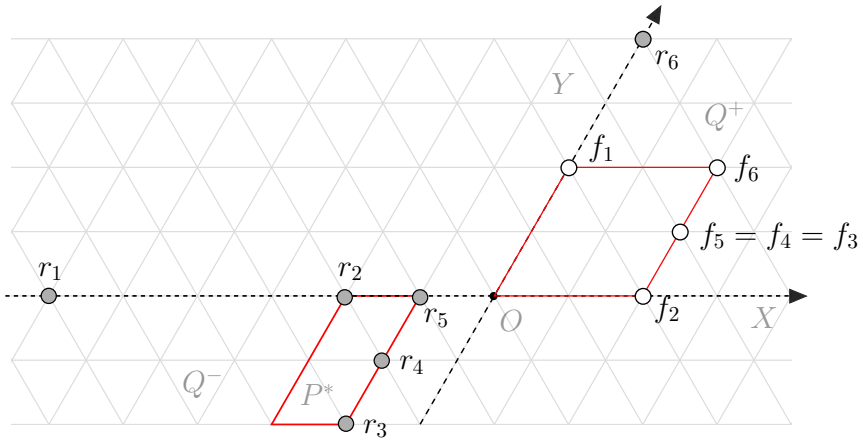


Figure 3.6: Visualization of Task T_3 is about the placing of guard $r_n = r_6$. Notice the embedding F_e in Q^+ (cf., Definition 3) of the pattern represented in Figure 3.4 and the ordering of all robots in R'' according to the lexicographic order of the coordinates of the vertices in which they reside.

let $\Delta = \max\{w(P^*), w(mbp(F))\}$.[1]

The target of r_1 is the closest vertex on the X-axis, at least at a distance 3Δ from the origin. The trajectory is any shortest path to the target. At the and of the task T_2, variable `g1` still holds and that the movement of r_1 makes the following additional variables, true:

- `hp''` = *all the robots in* R'' *are in the same half-plane with respect to the line induced by* `g1`.
- `dr1` $= d(r_1, O) \geq 3\Delta$.

Task T_3**.** This task brings r_n to a target recognizable in the subsequent tasks and during the partial pattern formation phase. As a precondition, we assume all the variables holding true at the end of the task T_2: `g1`, `hp''`, and `dr1`.

According to the precondition, robots can use Remark 2, but now the X-axis is directly defined as the direction induced by `g1`. Robots identify both guards and re-compute the common reference system. At this point, r_n performs the task by moving along the Y-axis (cf., Figure 3.6) toward the closest vertex $(0, y)$ such that the following variable holds:

- `gn` = r_n *is at a vertex* $(0, y)$*, with* $2\Delta \leq y < d(r_1, O)$.

[1]This definition of Δ is given for R'' instead of R' so that it is used in the subsequent tasks T_3, T_4, and T_5.

The next additional remark states how robots can re-compute the common reference system in subsequent tasks.

Remark 3. *At the end of Task T_3, i.e., when both the guards are in place, each robot can recognize the reference system: the two guards can be detected according to function $D()$, since r_1 and r_n have the largest and second largest value of $D()$, respectively; if* `g1` *holds, the induced line defines the X-axis directed from r_1 to all the other robots; the Y-axis is the line passing through r_n, intersecting the X-axis, directed from the intersection toward r_n, and forming a canonical angle in the first quadrant. Finally, the fact that the two guards are in place is verified relying on the value of Δ, since Q^- (which contains all robots in R'') is identified.*

Task T_4. This task solves the "Partial Pattern Formation" sub-problem. It forms the pattern F with robots in $R'' = R \setminus \{r_1, r_n\}$. All the $n-2$ robots in R'' initially located in the quadrant Q^- move in the quadrant Q^+. This task activates only when RS is completed. The precondition for task T_4 requires that `g1`, `gn`, and `dr1` are all true.

This task is accomplished only if the robots in R'' have a common reference system obtained as described in Remark 3. Then, all robots have to agree on the target positions in Q^+. To this aim, F is embedded into G_T according to the following definition.

Definition 3 (Embedding of the pattern)**.** *F_e is the set of vertices in Q^+ obtained by embedding F on the graph so that the following conditions hold: (1) the leading corner of $mbp(F)$ is mapped onto the origin O, and (2) the leading direction of $mbp(F)$ coincides with the positive direction of the Y-axis.*

An example of F_e is shown in Figure 3.6. Once the robots agree on F_e, the main difficulties in this task are to preserve the reference system (induced by guards r_1 and r_n) and to avoid undesired collisions during the movements. To avoid collisions, robots are moved one at a time according to a schedule induced by the following definitions:

- Vertices in F_e are ordered according to the lexicographic order of their coordinates expressed with respect to the formed X- and Y-axes. Hence, from now on we denote F_e as the multiset[2] $\{f_1, f_2, \ldots, f_n\}$, where $i \leq j$ if and only if the coordinates of f_i precede those of f_j. Similarly for robots in R'': they are ordered according to the lexicographic order of the coordinates of the vertices in which they reside and $R'' = \{r_2, r_3, \ldots, r_{n-1}\}$.
- Vertices f_1 and f_n are not used during the resolution of PPF since they are considered as the final *targets* for the guards. In particular, in the last part of the resolution algorithm, r_1 will be moved in f_1 and r_n will be moved in f_n.

[2]Recall that F may contain multiplicities.

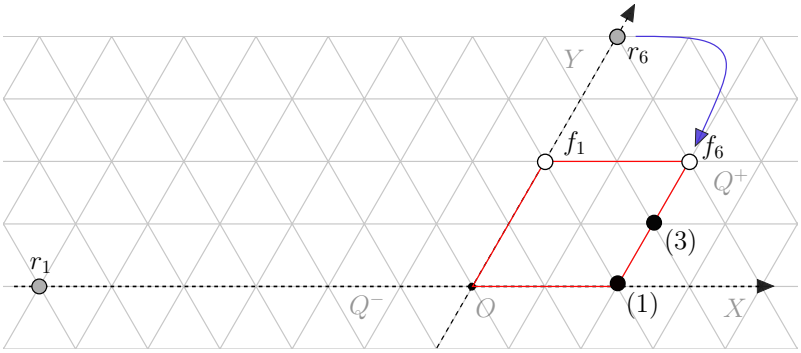


Figure 3.7: Visualization of the configuration at the end of task T_4 (cf., Figure 3.6). Gray (black, white, respectively) circles represent unmatched robots (matched robots, unmatched targets, respectively), while integers close to matched robots refer to multiplicities.

- A vertex $f_i \in F_e$, $2 \leq i \leq n-1$, is called the *largest unmatched target* if it is unoccupied whereas f_j is occupied for each $i < j < n$.
- A robot $r_i \in R''$, $2 \leq i \leq n-1$, is called *largest unmatched robot* if f_i is the largest unmatched target. Note that there is always a shortest path between r_i and f_i without other robots.

Algorithm $\mathcal{A}_{form}$ moves robots in R'' in order, moving each time the largest unmatched one toward the largest unmatched target in F_e. The trajectory of a moving robot is any shortest path leading to its target.

- `rpf` = *there exists a vertex v in Q^- such that the largest unmatched robot r_i is on a shortest path from v to f_i, and each robot r_j, $j < i$, is in Q^-.*

At the end of the task, variable `g1` still holds, and so also the following additional variables:

- `hp'` = *all the robots in R' are in the same half-plane with respect to the line induced by* `g1`*;*
- `pfn` = *there exists an embedding of F such that all robots in R'' are similar to $F \setminus \{f_1, f_n\}$.*

Task T_5. This task is the first associated with the "Finalization" sub-problem. In particular, it moves r_n toward f_n. All the variables holding at the end of the task T_4 are `g1`, `hp'`, `dr1`, and `pfn`.

Robot r_n moves from the Y-axis straightly along the canonical direction parallel to the X-axis until a vertex with the same X-coordinate of f_n is reached, and then

it directly proceeds toward the target (cf. Figure 3.7). The movement of r_n can take many LCM cycles therefore we define a new variable that checks the correct positioning of r_n:

- `hrn` = *point* $f_n = (x, y)$ *and robot* $r_n = (x', y')$, *with* $x' \leq x$ *and* $y' > y$.

Checking variable `hrn` requires a reference system. However, it is not calculable in the same way as in previous tasks since r_n is moving (i.e., guards are not in place anymore). Since `pfn` holds, the reference system can be deduced from the embedding. In particular:

Remark 4. During Task T_5, each robot can recognize the formed reference system: r_1 can be detected according to function $D()$ since it has the largest value of $D()$; if `g1` and `hp`$'$ hold, the induced line defines the X-axis directed from r_1 to all the other robots; from $mbp(F)$ it is possible to check whether its leading corner placed on a vertex v on the X-axis makes $f_2, \ldots, f_{n-1}$ matched, hence defining r_n; the Y-axis is assumed as the canonical direction passing through v and forming a canonical angle in the first quadrant which contains all robots in R'; finally, knowing r_1 and r_n and Q^- it is possible to compute Δ and hence check whether `dr1` holds.

Once r_n reaches f_n, variables `g1`, `dr1` still hold and a new variable becomes true:

- `pf1` = *pattern* $F \setminus \{f_1\}$ *formed.*

Tasks T_6 and T_7. When T_6 starts after the end of T_5, variables `g1`, `dr1`, and `pf1` are true. Therefore, $n-1$ robots reached their targets except for one robot, r_1, far enough from others to induce one direction toward the remaining robots. r_1 must finalize the pattern F by moving toward its target. During tasks T_6 and T_7, both guards are no longer in place so the common reference system is lost. In particular, the origin O of the system is not defined, and hence `dr1` cannot be evaluated. However, the algorithm moves r_1 so that the following variable remains valid during T_6:

- `dr1`$'$ = *the distance between* r_1 *and the other robots guarantees that* $d(r_1, mbp(F)) \geq 3w(mbp(F))$.

In particular, task T_6 moves r_1 toward a target vertex t so that the following properties hold: (1) `dr1`$'$ remains true, and (2) in task T_7, starting from t, robot r_1 reach its final target f_1 by moving straightly along one canonical direction.

Even though in both T_6 and T_7, the reference system is not available, robots can take advantage of the position of r_1 and of the other robots to finalize the pattern correctly. In particular, when T_6 starts, variables `g1`, `dr1`$'$, and `pf1` hold and, accordingly, robots can compute the following data:

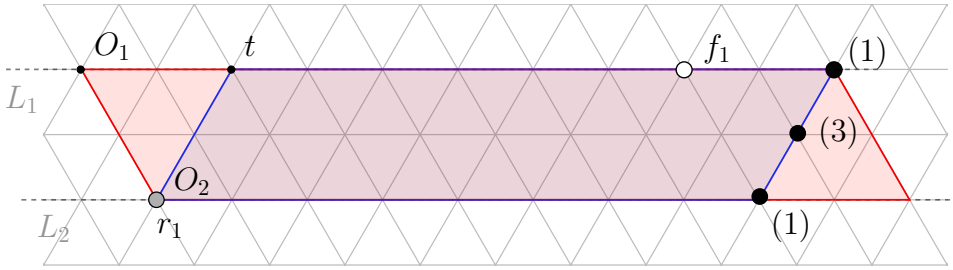


Figure 3.8: Visualization of the configuration at the end of task T_5 (cf., Figure 3.7). Gray (black, white, respectively) circles represent unmatched robots (matched robots, unmatched targets, respectively), while integers near matched robots refer to multiplicities.

- let U be the direction induced by variable `g1`. Let L_1 and L_2 be the lines parallel to U, closest to each other, and enclosing $R' = R \setminus \{r_1\}$;
- consider the smallest parallelogram P_1 (P_2, respectively) such that: encloses the whole set R, has the longest sides on L_1 and L_2, it admits the height equal to $h(mbp(F))$ and it determines a corner O_1 (O_2, respectively) at the intersection vertex with the shortest side that passes through r_1, that admits a canonical angle;
- compute sP_1 (sP_2, respectively) as the sequence of integers associated with O_1 (O_2 respectively) such that r_1 is met as the first robot.

For example, P_1 and P_2 correspond to the red and blue parallelograms shown in Figure 3.8, respectively. According to such data, robots verify whether the current configuration is coherent with task T_6 by performing the following check:

- at least one parallelogram between P_1 and P_2 must be coherent with the $n-1$ elements of F already matched. The last values of sP_1 (or sP_2) coincide with the sequence $\ell(mbp(F))$ except for one value corresponding to f_1 (the vertex to be matched by r_1). It is valid for both sP_1 and sP_2 when there is a reflection axis for $F \setminus \{f_1\}$ parallel to the direction U.

We denote by $\ell^F_{f_1}$ the sequence of integers obtained from $\ell(mbp(F))$ by decreasing by one the first non-zero element,[3] and by d_f the position in $\ell(mbp(F))$ of such an element. The check can be done by formalizing variable `pf1`:

- `pf1` = *there exists* $s \in \{sP_1, sP_2\}$ *such that* $s = s' + \ell^F_{f_1}$, *for some* s' *made of only* 0*'s and just one* 1 *in position* d_{r_1} *and* $d_{r_1} < d_f$.

[3] $\ell^F_{f_1}$ denotes the sequence $\ell(mbp(F))$ by ignoring f_1.

Referring to the example shown in Figure 3.8, variable `pf1` is made true by the sequence obtained from the vertex O_2. In fact, $sP_2 = (1, 0, 0,\ 0^{21},\ 1, 3, 1)$, $\ell(mbp(F)) = (0, 0, 1,\ 0, 0, 0,\ 1, 3, 1)$, $d_{r_1} = 1$ and $d_f = 3$.

During T_6, robot r_1 moves along the shortest side of the parallelogram associated with the string s (cf. the definition of `pf1`) to increase position d_{r_1}. The movement ends when $d_{r_1} = d_f$. When it happens,T_6 ends, and task T_7 begins and the following variable holds:

- `qf1` = *sequence* $\ell(mbp(R))$ *guarantees that* $\ell(mbp(R)) = \ell' + \ell^F_{f_1}$, *for some* ℓ' *made of only* 0*'s and just one* 1 *in position* d_{r_1} *and* $d_{r_1} = d_f$.

When `qf1` holds, all robots know that r_1 can complete the pattern by going straight toward its target. The difficulties arise from the possible symmetries formed during the last movement of r_1. The algorithm always produces asymmetric configurations during tasks $T_1, \ldots, T_6$, thanks to the positioning of r_1. When r_1 is very close to its target, symmetries may imply that more than one robot identifies itself as r_1, while r_1 can detect more than one vertex as its target f_1. However, we show that the configuration has at most a reflection axis, with r_1 on that axis, and does not prevent the pattern completion.

3.3.4 Algorithm formalization and correctness

As introduced in Section 3.2 , the proposed algorithm $\mathcal{A}_{form}$ is based on a strategy that decomposes the *APF* problem into tasks $T_1, T_2, \ldots, T_8$. All the needed basic variables useful for $\mathcal{A}_{form}$ have been already defined in Sections 3.3.3. If we assume that $\texttt{pre}_i$ is the composed variable that represents the pre-conditions of P_i, for each $1 \leq i \leq 8$, then predicate P_i can be defined as follow:

$$P_i = \texttt{pre}_i \wedge \neg(\texttt{pre}_{i+1} \vee \texttt{pre}_{i+2} \vee \ldots \vee \texttt{pre}_8). \tag{3.2}$$

This definition leads to the following remark:

Remark 5. *Predicates* P_i *fulfill Property* Prop_2. *This is directly implied by Eq. 3.2*

Before addressing the remaining properties Prop_1 and Prop_3, we formalize all the basic variables, the pre-conditions for each task, and, as a consequence, all the predicates. All the necessary basic variables are summarized in Table 3.1. Table 3.2 is organized as follows: the first two (general) columns refer to the hierarchical decomposition of the algorithm, the third column associates tasks names to sub-problems, and the fourth column defines precondition $\texttt{pre}_i$ for each task T_i. These preconditions are defined according to Equation 3.2. The fifth column of Table 3.2 contains the name of the move used in each task (we denote as m_i the move used in task T_i). Details for each move, are given in Table 3.3. Unless otherwise specified, each trajectory is any shortest path to the target.

var	*definition*	*rationale*
`g1`	$\exists$ a unique line parallel to a canonical direction passing through r_1 and each $bp(R')$	*guard r_1 is partially placed*
`gn`	r_n is at a vertex $(0, y)$, with $2\Delta \leq y < d(r_1, O)$	*guard r_n is placed*
`dr1`	$d(r_1, O) \geq 3\Delta$	*guard r_1 is at a desired distance from the origin*
`dr1'`	$d(r_1, mbp(F)) \geq 3w(mbp(F))$	*robot r_1 is at a desired distance from the pattern*
`hp'`	let L be the line induced by `g1`; all robots in R' are in the same half-plane with respect to L	*all robots in R' are in the same half-plane with respect to the line induced by* `g1`
`hp''`	let L be the line induced by `g1`; all robots in R'' are in the same half-plane with respect to L	*all robots in R'' are in the same half-plane with respect to the line induced by* `g1`
`hrn`	$f_n = (x, y)$ and $r_n = (x', y')$, with $x' \leq x$ and $y' > y$	*guard r_n is on the right path to its target*
`rpf`	$\exists$ a vertex v in Q^- such that the largest unmatched robot r_i is on a shortest path from v to f_i, and each robot r_j, $j < i$, is in Q^-.	*all the unmatched robots are correctly positioned with respect to PPF*
`pf1`	$\exists\ s \in \{sP_1, sP_2\}$: $s = s' + \ell^F_{f_1}$, for some s' made of only 0's and just one 1 in position d_{r_1} and $d_{r_1} < d_f$	*pattern $F \setminus \{f_1\}$ formed*
`pfn`	$\exists$ embedding of F such that all robots in R'' are similar to $F \setminus \{f_1, f_n\}$	*pattern $F \setminus \{f_1, f_n\}$ formed*
`qf1`	$\ell(mbp(R)) = \ell' + \ell^F_{f_1}$, for some ℓ' made of only 0's and just one 1 in position d_{r_1} and $d_{r_1} = d_f$	*guard r_1 can complete the pattern by going straight toward its target*
`s`	R and F are similar	*pattern F formed*

Table 3.1: The basic Boolean variables used to define all the tasks' preconditions.

problem	*sub-problem*		*task*	*precondition*	*move*
APF	*RS*	RS_{1a}	T_1	*true*	m_1
		RS_{1b}	T_2	$\mathtt{g1}$	m_2
		RS_2	T_3	$\mathtt{g1} \wedge \mathtt{hp}'' \wedge \mathtt{dr1}$	m_3
	PPF		T_4	$\mathtt{g1} \wedge \mathtt{dr1} \wedge \mathtt{gn} \wedge \mathtt{rpf}$	m_4
	Fin	Fin_1	T_5	$\mathtt{g1} \wedge \mathtt{hp}' \wedge \mathtt{dr1} \wedge \mathtt{hrn} \wedge \mathtt{pfn}$	m_5
		Fin_2	T_6	$\mathtt{g1} \wedge \mathtt{dr1}' \wedge \mathtt{pf1}$	m_6
		Fin_3	T_7	$\mathtt{qf1}$	m_7
	Term		T_8	$\mathtt{s}$	*nil*

Table 3.2: Algorithm $\mathcal{A}_{form}$ for APF. The algorithm works as follows: if if a robot detects that predicate P_i holds, (where P_i depends on preconditions as defined in Eq. 3.2), it recognizes that task T_i must be performed and hence performs move m_i.

Table 3.2 leads to the following remark:

Remark 6. *Algorithm $\mathcal{A}_{form}$ fulfills Property Prop_3. This is implied by pre-condition $\mathtt{pre}_1$ and predicates P_i.*

Computability of the predicates: property Prop_1

In this section, we explain how algorithm $\mathcal{A}_{form}$ can compute each predicate P_i, showing that property Prop_1 holds.

According to the definition of P_i given in Eq. 3.2, in the `Compute` phase, each robot evaluates predicates (for the perceived configuration C and the pattern F) starting from P_8, in reverse order until it finds a true pre-condition. In case all pre-conditions $\mathtt{pre}_8, \mathtt{pre}_7, \ldots, \mathtt{pre}_2$ are evaluated false, then task P_1 is performed.

$\mathtt{pre}_8$ require checking whether C and F are similar. For the evaluation of $\mathtt{pre}_7$, robots need to compute variable `qf1`, which depends on $mbp(R)$ and $mbp(F)$. Pre-condition $\mathtt{pre}_6$ require the computation of `g1`, $\mathtt{dr1}'$, and `pf1`. `g1` requires r_1, identified according to function $D()$. In all tasks, $T_1, \ldots, T_6$, r_1 corresponds to the robot r such that $D(r)$ is maximum. $\mathtt{dr1}'$ require r_1 and $mbp(F)$ and `pf1` is computable as shown in Section 3.3.3 starting from the direction U induced by variable `g1`, U and $mbp(F)$ to determine the sequences sP_1 and sP_2, and the positions d_{r_1} and d_f.

Pre-condition $\mathtt{pre}_5 = \mathtt{g1} \wedge \mathtt{hp}' \wedge \mathtt{dr1} \wedge \mathtt{hrn} \wedge \mathtt{pfn}$ is evaluated as follows: `g1` can be evaluated once r_1 is recognized with the function $D()$, and $\mathtt{hp}'$ can be detected once the direction induced by variable `g1` is known. Now, as described in Remark 4, by using `g1` and $\mathtt{hp}'$, the common reference system can be recognized by each robot and

move	*definition*
m_1	r_1 moves toward the closest vertex so as `g1` holds
m_2	r_1 moves toward the closest vertex on X-axis at distance at least 3Δ from the origin
m_3	r_n moves toward vertex $(0, y)$, with $2\Delta \leq y < d(r_1, O)$
m_4	the *largest unmatched robot* in R'' moves toward the *largest unmatched target* in F_e
m_5	r_n first moves along a path that maintains fixed the y coordinate until its x coordinate coincides with that of f_n - then, it moves toward f_n
m_6	if both sP_1 and sP_2 satisfy `pf1` then let s be the lexicographically minimum one. Then, robot r_1 moves along the shortest side of the parallelogram associated with s so as to increase d_{r_1}
m_7	robot r_1 in position d_{r_1} moves toward f_1

Table 3.3: Moves associated to tasks. It is assumed that each robot not involved in m_i performs the *nil* movement.

with it, both `dr1` and `hrn` can be evaluated. Finally, variable `pfn` can be checked with a combinatorial approach.

All variables of pre-condition $\texttt{pre}_4 = \texttt{g1} \wedge \texttt{dr1} \wedge \texttt{gn} \wedge \texttt{rpf}$, except for `rpf` can be evaluated according to Remark 3. `rpf` can be verified as stated by the definitions introduced in Section 3.3.3. Pre-condition $\texttt{pre}_3 = \texttt{g1} \wedge \texttt{hp}'' \wedge \texttt{dr1}$ is evaluated as follows: `g1` is computed with the function $D()$, then Remark 2 is used to recognize the common reference system. Given the reference system, both $\texttt{hp}''$ and `dr1` can be evaluated. Finally, to check $\texttt{pre}_2 = \texttt{g1}$, robot r_1 is identified with the function $D()$.

Correctness

In this section, we formally prove that algorithm $\mathcal{A}_{form}$ solves the APF problem on the tessellation graph G_T. To this end, let $\mathcal{I}_{\mathcal{A}}$ be the set containing all the configurations taken as input or generated by $\mathcal{A}_{form}$.

According to properties Prop2 and Prop3, all tasks' predicates P_1, P_2, ..., P_8 used by the algorithm are defined to make a partition of $\mathcal{I}_{\mathcal{A}}$. Together with Prop_1, for each possible configuration provided to $\mathcal{A}_{form}$, the algorithm can evaluate each predicate and determine the task to perform.

Correctness can be assessed by proving that all the following properties hold:

H_1**:** $\mathcal{A}_{form}$ does not generate multiplicities nor symmetric configurations (unless F is formed or its formation is not prevented);

H_2**:** from any class T_i, $2 \leq i \leq 8$, no class T_j with $j < i$ can be reached;

H_3**:** from any class T_i, $1 \leq i \leq 7$, another class T_j with $j > i$ is always reached within a finite number of LCM cycles.

Since properties H_1, H_2 and H_3 must be proved for each transition/move, then in the following we provide a specific lemma for each task.

Lemma 1. *From an initial configuration C belonging to class $T_1 \cap \mathcal{I}_{\mathcal{A}}$ the algorithm $\mathcal{A}_{form}$ eventually leads to a configuration C' in a class T_i, $i > 1$.*

Proof. In this task, algorithm $\mathcal{A}_{form}$ selects a robot, denoted as r_1 (the first guard), such that $D(r_1)$ is maximum and, in case of ties, the robot that has the minimum position in $\ell(mbp(R))$.

(*Property* H_1). Since $D(r_1)$ is maximum, while r_1 moves away from the other robots, it cannot meet any other robot and $D(r_1)$ increases. Then, r_1 is repeatedly selected. Note that, if by m_1 a symmetric configuration is created then it must admit an axis of reflection where r_1 lies – as this is the only robot defining $D(r_1)$.

(*Property* H_2). Since as we are going to show the subsequent H_3 holds, we have that any other class can be reached.

(*Property* H_3). Robot r_1 always decreases the distance toward its target, within a finite number of LCM cycles, unless other predicates become true, g1 becomes true and the configuration is not in T_1 anymore. □

Lemma 2. *From a configuration C belonging to class $T_2 \cap \mathcal{I}_{\mathcal{A}}$ the algorithm $\mathcal{A}_{form}$ eventually leads to a configuration C' in a class T_i, $i > 2$.*

Proof. Here r_1 lies between two parallel directions L_1 and L_2 enclosing each possible $bp(R')$ and moves toward the closest one (toward any of them in case of ties) along a canonical direction.

(*Property* H_1). Robot r_1, when moving toward its target, cannot meet any other robot, nor move on any axis of symmetry because the only possible one should be at the same distance from L_1 and L_2 and parallel to them. However, by moving to the closest L_i, $i \in \{1, 2\}$, r_1 never crosses an axis.

(*Property* H_2). Move m_2 does not affect predicate g1, that is no obtained configuration can belong to T_1.

(*Property* H_3). Robot r_1 always decreases the distance toward L_i, then within a finite number of LCM cycles, unless other predicates become true, $\mathtt{hp}'' \wedge \mathtt{dr1}$ becomes true (cf. Section 3.3.3) and the configuration is not in T_2 anymore. □

Lemma 3. *From a configuration C belonging to class $T_3 \cap \mathcal{I}_{\mathcal{A}}$ the algorithm $\mathcal{A}_{form}$ eventually leads to an asymmetric configuration C' in T_i, $i > 3$.*

Proof. During this task, guard r_1 is already placed, that is $\texttt{g1} \wedge \texttt{hp}'' \wedge \texttt{dr1}$ holds.

(*Property* H_1). Due to the positioning of r_1, the configuration can be symmetric only when all robots are collinear (along the formed X-axis). Regardless of when this symmetry is formed, during this task, r_n is always detectable. When it leaves the X-axis, the configuration becomes asymmetric and remains so until the second guard ends its trajectory. According to m_3, along with its movement r_n does not meet any other robot. Since the distances from O of the two guards are different, the configuration cannot admit rotations or reflections as long as the guards are idle.

(*Property* H_2). Move m_3 does not affect predicates `g1`, $\texttt{hp}''$ and `dr1`, therefore any obtained configuration cannot belong to T_1 nor to T_2.

(*Property* H_3). Robot r_n always decreases the distance toward its target along the Y-axis, then within a finite number of `LCM` cycles, unless other predicates become true, `gn` becomes true. In any case, the configuration is not in T_3 anymore. □

Lemma 4. *From any configuration C belonging to class $T_4 \cap \mathcal{I}_\mathcal{A}$ the algorithm $\mathcal{A}_{form}$ eventually leads to a configuration C' in T_5.*

Proof. (*Property* H_1). Since guards r_1 and r_n are placed, the same considerations of Lemma 3 hold, that is the configuration cannot admit reflections nor rotations during this task. Multiplicities can be created but only if required by F.

(*Property* H_2). During the whole task, predicate `s` is false as guards remain placed. Hence, also predicates `g1`, `dr1`, `gn` and `rpf` are not affected by m_4, that is the obtained configuration cannot belong to T_1, T_2, and T_3.

(*Property* H_2). While the task is performed, either the number of matched robots increases or the distance of one robot from its target decreases, then in a finite number of moves all robots excluding r_1 and r_n will be matched. As already described in Section 3.3.3, at the end of this task $\texttt{g1} \wedge \texttt{hp}' \wedge \texttt{dr1} \wedge \texttt{hrn} \wedge \texttt{pfn}$ holds, that is C' belongs to T_5 and no other task can be reached because the guards remain placed. □

Lemma 5. *From any configuration C belonging to class $T_5 \cap \mathcal{I}_\mathcal{A}$ the algorithm $\mathcal{A}_{form}$ eventually leads to a configuration C' in T_i, $i > 5$.*

Proof. During this task, guard r_1 remains placed.

(*Property* H_1). As r_n moves toward its final target, the arisen configurations cannot admit reflections nor rotations as there are no other robots equivalent to r_1 due to `g1` and `dr1`. A reflection (as well as a multiplicity, respectively) can occur only at the end of the task if all robots are collinear (if f_n requires a multiplicity, respectively) but this can be managed by $\mathcal{A}_{form}$ as we are going to see in the next lemma devoted to T_6.

(*Property* H_2). Before r_n reaches its target, $\texttt{g1} \wedge \texttt{hp}' \wedge \texttt{dr1} \wedge \texttt{hrn} \wedge \texttt{pfn}$ remains true, while predicates `pf1` and `s` remain false, that is the configuration remains in T_5. Once r_n reaches f_n, predicate $\texttt{g1} \wedge \texttt{dr1}' \wedge \texttt{pf1}$ becomes true.

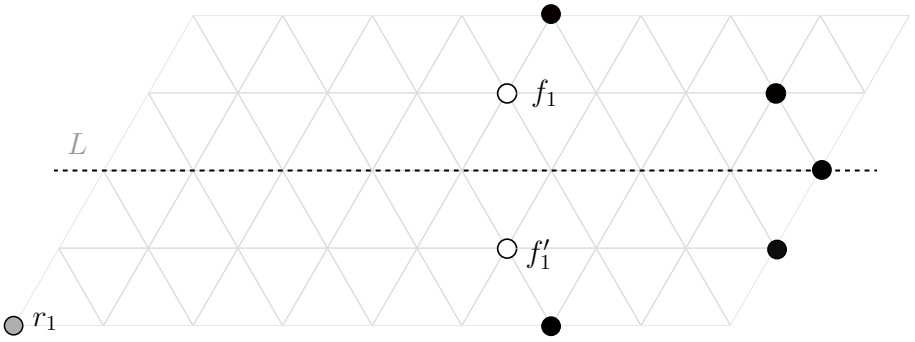


Figure 3.9: An example of the only possible symmetry that can arise during task T_6.

(*Property* H_3). After each move, r_n decreases its distance from f_n, that is within a finite number of `LCM` cycles the task ends and, unless other predicates become true, the obtained configuration C' belongs to T_6. □

Lemma 6. *From a configuration C belonging to class $T_6 \cap \mathcal{I}_\mathcal{A}$ the algorithm $\mathcal{A}_{form}$ eventually leads to a configuration C' in T_7.*

Proof. During this task, guard r_1 moves so as to make $d_{r_1} = d_{f_1}$.

(*Property* H_1). During this phase, the algorithm does not generate any multiplicity since $\texttt{g1} \wedge \texttt{dr1}' \wedge \texttt{pf1}$ remains true and then r_1 is sufficiently far from any other robot. Regarding symmetries (cf. Figure 3.9), the only symmetric configuration possible is the one with an axis parallel to the direction induced by `g1`, and f_1 can be on the axis or not. In the first case, the whole pattern is symmetric; when r_1 reaches the axis then $d_{r_1} = d_{f_1}$ holds and predicate `qf1` becomes true. Otherwise the final pattern is asymmetric and there are two possible embeddings and two possible targets for r_1, f_1 and its equivalent point f_1' with respect to the axis of symmetry. One of the two is reachable by r_1 without crossing the axis. Consider the two half planes determined by the line that makes `g1` true: the targets f_1 and f_1' may lie in the same half plane or not. In the first case only one among the sequences sP_1 and sP_2 satisfies the condition in `pf1` because in one of them $d_{r_1} > d_{f_1}$. Then r_1 moves towards f_1 and when it reaches the height of f_1' predicate `qf1` becomes true and the configuration is in T_7. When r_1 lies between f_1 and f_1', both the sequences sP_1 and sP_2 satisfy the condition in `pf1` and move m_6 chooses the smaller one since sP_1 and sP_2 must be different because r_1 is not on the axis. Robot r_1 increases its height to align with the target until $d_{r_1} = d_{f_1}$ and the configuration is in T_7.

(*Property* H_2). Before r_1 reaches the height of its target, $\texttt{g1} \wedge \texttt{dr1}' \wedge \texttt{pf1}$ remains true, and when $d_{r_1} = d_{f_1}$, `qf1` holds, hence C' is in T_7. Clearly C' cannot belong to T_8.

(*Property* H_3). The absolute difference between d_r and d_f decreases by one at each move until $d_{r_1} = d_{f_1}$ so that move m_6 is applied only a finite number of times. □

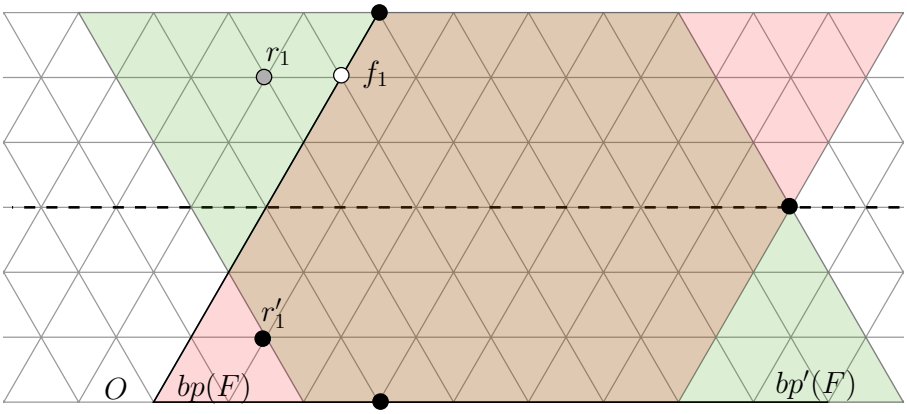


Figure 3.10: An example in which r_1 becomes equivalent to another robot r'_1 respect to an axis of 0° while moving toward f_1.

Lemma 7. *From a configuration C belonging to class $T_7 \cap \mathcal{I}_{\mathcal{A}}$ the algorithm $\mathcal{A}_{form}$ eventually leads to a configuration C' in T_8.*

Proof. During this task, guard r_1 straightly moves toward its target. Since `qf1` holds it is possible to derive the embedding of the pattern from $\ell(mbp(F))$ and consequently the X and Y axis that we refer to the proof.

(*Property H_1*). We show that while r_1 moves toward f_1 no reflections, no rotations, no multiplicities can be created that prevent the finalization of the task. In particular, we first show no reflection can admit a robot equivalent to r_1 (hence, if a reflection is created, then r_1 must be on the axis of symmetry, and we show this happens only if $F \setminus \{f_1\}$ is symmetric respect to that axis). Then, we show that no rotations are possible. Finally, concerning the multiplicities, r_1 can make one only once f_1 is reached.

About reflections. Regarding reflections we have to analyze possible axis of reflection at 0°, 30°, 60°, 90°, 120°, 150°, with respect to the X-axis in clockwise direction. Moreover, we distinguish between two cases: when r_1 becomes equivalent to another robot of the configuration and when r_1 goes on an axis of symmetry.

Firstly, we analyze the case of a reflection at 0° when r_1 becomes equivalent to another robot r'_1 while moving toward f_1 (see Figure 3.10). Now consider the other possible $bp'(F)$ having two sides parallel to the X-axis and shared with the chosen $bp(F)$. One side of $bp'(F)$ passes through r'_1 and the reading from this side is lower than the reading of $bp(F)$ from the origin. Then the embedding chosen was not coherent with the definition, a contradiction.

For the cases of reflections at 30° and 60° the supposed robot r'_1 equivalent to r_1, would lie outside the embedding of $mbp(F)$.

Regarding the case of a reflection at 90°, that is a reflection perpendicular to the X-axis, r_1 becomes equivalent to another robot r'_1 while moving toward its target

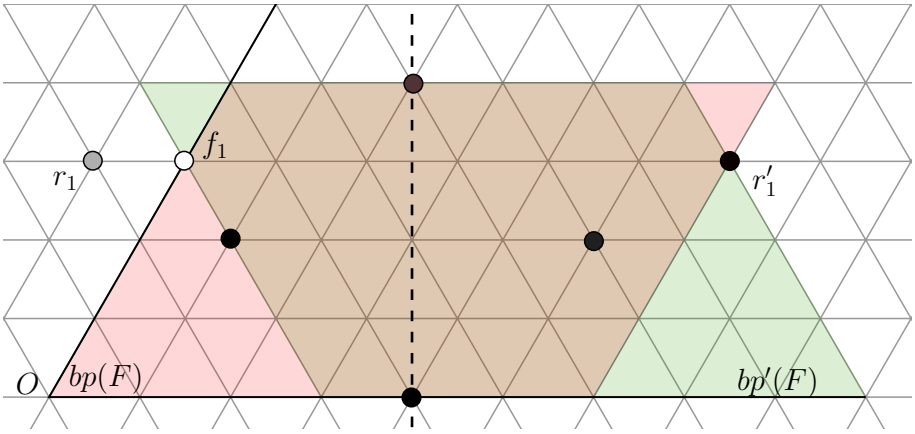


Figure 3.11: An example in which r_1 becomes equivalent to another robot r'_1 respect to an axis of 90° while moving toward f_1.

(see Figure 3.11). Now consider the other possible $bp'(F)$ having two sides parallel to the X-axis and shared with the chosen $bp(F)$. As in the case of a reflection at 0°, one side of $bp'(F)$ passes through r'_1 and the reading from this side is lower than the reading of $bp(F)$ from the origin. Then the embedding chosen was not coherent with the definition, a contradiction.

Regarding the case of a reflection at 120°, the reflectional axis is parallel to the Y-axis, and r_1 becomes equivalent to another robot r'_1 while moving toward its target. The axis of symmetry must be between O and the half of the longest side of $bp(F)$. We now compare the reading of $bp(F)$ from O with the reading of $bp(F)$ starting from the corner at the opposite angle of 60° respect to O, call it P. The first column read from P has at most one robot, r'_1 equivalent to r_1, then as many empty columns as those found from r_1 to the Y-axis in $mpb(R)$, until a first robot specular to the one read from O. Since the number of empty columns read from P is greater than the one read from O, the reading from P is lower than the reading from O hence a contradiction.

In case of a reflection axis at 150°, the Y-axis reflects on the X-axis, then there is no possible robot r'_1 in the configuration that can be equivalent to r_1 when approaching to its target.

In what follows, we analyze the case when $R \setminus \{r_1\}$ forms an axis of symmetry.

Consider the case of a reflection at 0°. If the pattern is symmetric respect to that axis, f_1 is on the axis, and r_1 reaches the axis and proceeds along the axis without breaking the symmetry, by following the trajectory specified by move m_7. If the pattern is asymmetric, then there are two possible embedding of F on $R \setminus \{r_1\}$ and then there must be another target f'_1 equivalent to f_1 obtained by reflecting the embedding such that the trajectory computed by the move of r_1 does not cross the axis (see Lemma 6). According to move m_7, actually robot r_1 moves to f'_1 to finalize the task.

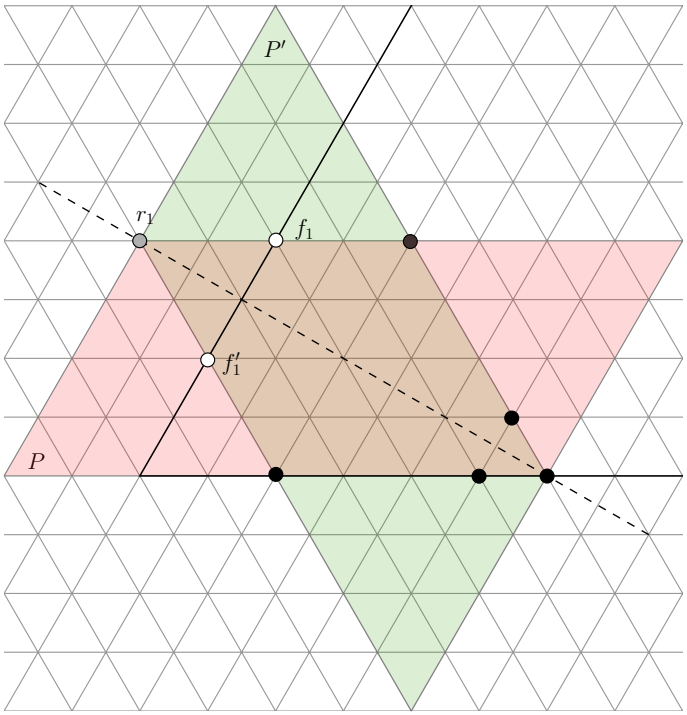


Figure 3.12: Robot r_1 on a reflection axis of 30° and the equivalent parallelograms P and P'.

In case of a reflection axis of 30°, r_1 goes towards that axis and when it lands on it there two equivalent parallelograms $P = mbp(R)$ and its reflection P' (see Figure 3.12). Let $l(P)$ and $l(P')$ the readings of the two parallelograms. These sequences are equivalent and they both find r_1 as the first robot. In each sequence r_1 is univocally determined and it can move respect to either P or P' toward f_1 or f_1', respectively. As r_1 moves away from the axis, there is a unique $mbp(R)$ until r_1 reaches its target.

It is easy to see that when moving r_1 cannot go on an axis of 60°, 90°, and 120° before reaching its target.

Regarding to axes of 150°, r_1 could go on such an axis only if f_1 is under the reflection axis, but to be symmetric with such an axis the pattern should have the longest side laying on the Y-axis and this is not coherent with the embedding.

About rotations. The minimal possible angle of rotation is 60° and its multiples 120° and 180°, clockwise and anti-clockwise. The convex hull of any configuration with rotational symmetry with angle of rotation of 60° is an hexagon. Assuming that such a configuration is formed when r_1 is approaching its target, a part of the convex hull should be in the quadrant where r_1 lies. Then the embedding of the pattern is not positioned according to the rule that the shorter side of the parallelogram is parallel to the Y-axis (cf., Definition 3). With the same arguments we can exclude

rotations of 120°. Regarding to rotations of 180°, let us assume that r_1 creates such a symmetry when approaching its target. The embedding F_e is done by construction in such a way that the sequence of integers read from the origin is smaller than the one read from the corner $P = (x_p, y_p)$ at the opposite angle of 60°. The first column read from P must have a single robot r'_1, symmetric to r_1, because this column matches the one with r_1.

By hypothesis, the pattern sequence read from O must be lower than the one read from P, then the first column cannot have more than one target and in particular this target must be at the same distance from O than r'_1 from P, because r_1 is moving horizontally. Reading the configuration forward from P, there must be a sequence of columns of zeros, at least one, each corresponding to an empty column read from r_1 to the Y-axis, that is empty. In turn, this corresponds to a sequence of columns of zeros in the pattern read from O, because by hypothesis must be lower than the one read from P. Then, by rotation, these columns correspond to more empty columns in the configuration read from P. Continuing, we would have only empty columns between r_1 and r'_1, contradicting the hypothesis that the robots are at least three.

In conclusion, when moving r_1 does not create any rotation or reflection with a robot becoming equivalent to r_1. The two cases in which r_1 creates a symmetric configuration is when it is on an horizontal axis and it moves along that axis or when is on a 30° axis and in this situation r_1 can always break the symmetry.

To conclude the proof of H_1, we also need to ensure that r_1 is always recognized until reaching f_1. In fact, as long as r_1 is sufficiently far away from the other robots it is easily recognizable according to its distance from O. When r_1 is close to the other robots is still always recognizable. In fact the parallelogram $mbp(R)$ is unique (apart from the case in which r_1 is on an axis of symmetry at 0° and 30°) and it can't be a square due the position of r_1 then there are two sequences of integers associated to the canonical corners of the $mbp(R)$. The minimal one finds r_1 as the first robot; in fact if there were another robot playing the role of r_1 in the minimal reading that reading would be a palindrome to the first sequence and that means that the configuration is symmetric. Since the algorithm doesn't create symmetric configurations, such palindrome reading cannot exists and then r_1 is unique. If r_1 lies on an axis, there are two parallelograms equivalent to $mbp(R)$ but the sequence of integers associated with these parallelograms finds r_1 as the first robot, then again r_1 is uniquely identified.

(*Property* H_2). During the movement of r_1, predicate `qf1` remains true because $n-1$ robots are already matched, they all stay still and r_1 straightly moves towards its target along the direction of the longest side of $mbp(F)$. This implies that the sequence $\ell(mbp(R))$ keeps its structure given by the concatenation of a subsequence ℓ' made of only 0s and just one 1 in position d_{r_1} and a subsequence $\ell^F_{f_1}$ that encodes the position of the robots already matched. When r_1 reaches its target $\ell(mbp(R)) = \ell(mbp(F))$ and the configuration is in T_8.

(*Property* H_3). After each move, r_1 decreases the distance from f_1 while the sequence ℓ' gets smaller by a number of 0s equal to the shorter side of $mbp(R)$ until $\ell(mbp(R)) = \ell(mbp(F))$. This implies that within a finite number of `LCM` cycles `s` becomes true and C' belongs to T_8. □

Remark 7. *We have shown that in fact algorithm* $\mathcal{A}_{form}$ *manages not only asymmetric configurations but also some leader configurations where only one robot has to move and it is recognizable as one of the two guards* r_1 *or* r_n*.*

Theorem 1 (Correctness)**.** *Let* $C = (G_T, \lambda)$ *be any initial configuration with* $n \geq 3$ ASYNC *robots, and let* F *be any pattern (possibly with multiplicities) such that* $|F| = n$*. Then,* $\mathcal{A}_{form}$ *is able to form* F *starting from* C*.*

Proof. What we are going to show is that if all three properties H_1, H_2 and H_3 hold, then for each possible execution of $\mathcal{A}_{form}$ there exists a time t^* such that $C(t^*)$ is similar to F and $C(t) = C(t^*)$ for any time $t \geq t^*$. This implies that the statement holds.

Assume that C is provided as input to $\mathcal{A}_{form}$. According to properties Prop_1, Prop_2 and Prop_3, there exists a single task (say T_i) to be assigned to robots with respect to C. According to H_1, any configuration generated from T_i (say C') can be provided as input to $\mathcal{A}_{form}$. Moreover, by H_2 and H_3, we can consider C' belonging to some class (say T_j) different from T_i. According to this analysis, we can say that C' will evolve during the time by changing its membership from class to class according to the forward transitions defined by Lemmas 1–6. Although the execution of $\mathcal{A}_{form}$ is infinite, property H_3 assures that any task is completed within a finite number of `LCM` cycles, apart for T_8 that will be reached within finite time t^*. Moreover, as the only movement allowed in T_8 is the *nil* one, then the reached configuration will not change anymore. □

3.3.5 Algorithm extension to graphs G_S and G_H

In this section, we briefly discuss how algorithm $\mathcal{A}_{form}$ can be extended to solve the APF problem for asymmetric configurations defined on G_S or G_H.

Our algorithm uses a few geometric concepts, such as bounding parallelogram, grid line, shortest path, moving along a line, and quadrant. Moving from G_T to G_S all these concepts remain valid and the canonical directions reduce to two, and consequently $bp(R)$ is unique. Moves do not need any changes. Since predicates are independent of the underlying graph, there is no need to change them. Hence the algorithm $\mathcal{A}_{form}$ remains the same, and the proof of correctness still holds while considering the variations due to the reduction of the canonical directions.

Moving to hexagonal grids, G_H is considered as a subgraph of G_T in which the center of the hexagons corresponds to removed vertices. By assuming the "presence" of the missing nodes and edges in relation to G_T, most of the geometric concepts

introduced are still valid except for "movement along a line". A robot cannot move along a line but it needs to move along the edges of successive hexagons. For instance, in tasks T_1 and T_2, $\mathcal{A}_{form}$ requires that r_1 reaches the target via shortest paths, without assuming other constraints. So, even in G_H the moves m_1 and m_2 remain valid. Conversely, during T_3, r_n moves along the Y-axis according to $\mathcal{A}_{form}$. In this case, we need to specify how a similar movement can be realized since there are missing edges of G_T. In the next subsection, we revise the algorithm and give the details of the changes needed to extend $\mathcal{A}_{form}$ to hexagonal grids.

Hexagonal grid graphs

G_H is considered as a sub graph of G_T in which the center of the hexagons corresponds to removed vertices. The basic concepts defined for G_T naturally extend to G_H. In particular:

- the distance function between two vertices u and v in G_H is the length of a shortest path connecting u and v in G_T;
- canonical directions in G_H are the directions of the edges incident to a single vertex, the same introduced in G_T. Given the canonical directions, we consider the same definition for a *mpb* as in G_T. Given a vertex v and an oriented line L passing through v toward a canonical direction, vertex v can be classified in one of these three types:
  - type 0: if v is not in G_H;
  - type 1: if v has an edge following the orientation of L;
  - type 2: otherwise.

  The type of a leading corner is determined by the reading in the same direction that originates the sequence of $mbp(F)$.
- the sequence of integers associated to a configuration of robots is the same as defined for G_T placing a zero in the sequence in correspondence of a vertex in G_T but not in G_H.

Further concepts will be introduced in the following description of the algorithm. In the hexagonal graph, to go toward a direction, a robot either moves to the adjacent vertex if there is an edge connecting the two or it moves along the edges of the next hexagon ahead. Therefore a robot moves alternatively straight or diverting its path. As a result, the movement of a robot is enclosed in a band that is tall half the height of an hexagon while moving toward a direction (cf. Figure 3.13). Given a robot and three canonical directions, there are two bands for each direction, the band selected each time by the robot is specified in a task when needed.

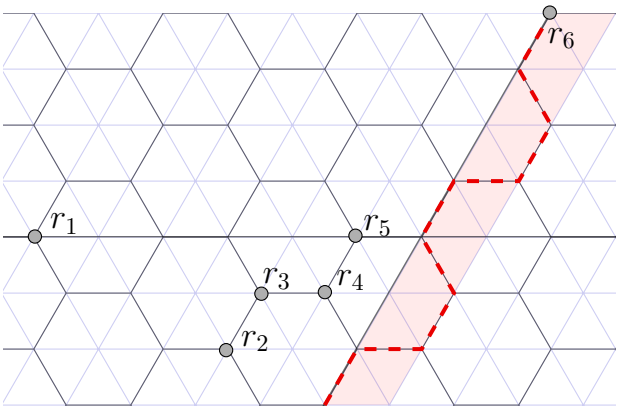


Figure 3.13: Robot r_6 moving in a band during task T_3.

- Task T_1: During this task, robot r_1 moves away from the other robots until predicate `g1` becomes true. For hexagonal grids predicate `g1` is updated as follows:
  `g1` : r_1 is at a vertex such that exists a unique direction in which at least one of the lines passing through r_1 or one of its neighbors encounters each $bp(R')$.

- Task T_2: In this task, r_1 moves at a distance 3Δ from the origin. The origin here is redefined since it can be a vertex of G_T not in G_H. Given r_1 and r_n, let R'' be $R'' = R \setminus \{r_1, r_n\}$. Let L be the line that forms a canonical angle with X passing through a robot in R'' and farthest from r_1. The origin is defined as the first vertex encountered from the intersection of L and the X-axis, having the same type of the leading corner of $mbp(F)$ read by following the orientation of the Y-axis.

- Task T_3: In this task, r_n moves toward its target through any shortest path while keeping outside $mpb(R'')$ also during a detour.

- Task T_4: In this task, $n-2$ robots reach their target one by one. This task develops in the same way as in G_T.

- Task T_5: In this task, guard r_n goes towards its target f_n. While moving parallel to the X-axis, r_n moves in any band that keeps at least 2Δ distance from the X-axis. While moving parallel to the Y-axis, r_n moves in the band farthest from r_1. Predicate `hrn` is updated as follows:
  `hrn` : $f_n = (x, y)$ and $r_n = (x', y')$, with $x' \leq x + 1$ and $y' > y$

- Task T_6: In this task, r_1 moves parallel to the shortest side of the parallelogram $mbp(R)$ as to increase d_r. During the movement r_1 moves in any band that keeps at least $3\texttt{Mf}(mbp(F))$ distance from $mbp(F)$. We say that r_1 is in line with its target if $d_r = d_f$ or $d_r - 1 = d_f$ since r_1 is moving within a band, so predicate `qf1` updates as follows:

`qf1` : $\ell(mbp(R)) = \ell' + \ell^F_{f_1}$, for some ℓ' made of only 0's and just one 1 in position d_r and $(d_r = d_f \vee d_r - 1 = d_f)$.

- Task T_7: In this task, r_1 moves towards its target and in case of detours it moves in the direction such that $\ell(mbp(R))$ decreases.

The same proofs of correctness given in Section 3.3.4 for G_T apply for G_H.

3.3.6 Concluding remarks

We provided an algorithm that solves the *APF* problem for asynchronous robots moving on any regular tessellation graphs (i.e., triangular, square, and hexagonal grids) as a discretization of the Euclidean plane. As a relevant improvement, compared to previous results, our algorithm works for any tessellation graph and also admits patterns with multiplicities. As a possible limitation, our algorithm assumes only asymmetric configurations as input.

For robots moving in the Euclidean plane, *APF* can be solved if and only if a leader configuration is provided as input [38]. Leader configurations form a superset of all the asymmetric configurations since they are defined as those configurations in which it is possible to elect a leader robot. They contain symmetric configurations in which a leader robot is located in the center of symmetry or on an axis of symmetry. In the Euclidean plane, the robots are assumed to be able to execute accurate movements in any direction and by any amount, even by infinitesimally small amounts. Therefore, even in densely crowded situations, a leader robot can always maneuver to leave the center or the axis, and thus break the symmetry. Of course, in graphs this simple strategy cannot be applied as the movements of the robots are restricted to the neighborhood (e.g., consider the case of a rotational configuration defined on G_S with a robot on the center of rotation and all its four neighbors occupied). Hence, it may happen that the leader cannot move without causing a multiplicity which might prevent the formation of the final pattern. As a consequence, before moving the leader, a resolution strategy should make "enough space" around the leader. This topic has been recently investigated in [33] for configurations defined on G_S or G_T. As a natural extension of our work, it would be interesting to check whether the strategy proposed in [33] can be combined with $\mathcal{A}_{form}$. If possible, it would characterize the *APF* problem on both G_S and G_T. However, from a first attempt, the composition of the two algorithms is not straightforward, mainly due to the occurrence of possible pending moves.

3.4 The Geodesic mutual visibility problem on trees

One of the basic tasks for mobile robots, intended as points in the plane, is certainly the requirement to achieve a placement so as no three of them are collinear. Furthermore, during the whole process, no two robots must occupy the same position concurrently, i.e., *collisions* must be avoided. This is known as the MUTUAL VISIBILITY problem. The idea is that, if three robots are collinear, the one in the middle may obstruct the reciprocal visibility of the other two.

Mutual Visibility has been largely investigated in recent years in many forms, subject to different assumptions. We introduce the GEODESIC MUTUAL VISIBILITY problem (GMV, for short): starting from a configuration composed of robots located on distinct vertices of an arbitrary graph, within finite time the robots must reach, without collisions, a configuration where they all are in geodesic mutual visibility. Robots are in geodesic mutual visibility if they are pairwise mutually visible, and two robots on a graph are mutually visible if there is a shortest path (i.e., a "geodesic") between them along which no other robots reside. This new problem can be thought of as a possible counterpart to the MUTUAL VISIBILITY for robots moving in a discrete environment. While this concept is interesting by itself, its study is motivated by the fact that robots, after reaching a GMV condition, e.g., can communicate in an efficient and "confidential" way, by exchanging messages through the vertices of the graph that do not pass through vertices occupied by other robots or can reach any other robot along a shortest path without collision. Concerning the last motivation, in [15] it is studied the COMPLETE VISITABILITY problem of repositioning a given number of robots on the vertices of a graph so that each robot has a path to all others without visiting an intermediate vertex occupied by any other robot. In that work, the required paths are not the shortest paths and the studied graphs are restricted to infinite square grids and infinite hexagonal grids, both embedded in the Euclidean plane. Recently, the geodesic mutual visibility has been investigated in [62] from a pure graph-theoretical point of view in order to understand how many robots, at most, can potentially be placed within a graph G in order to guarantee GMV. Such a number of robots has been denoted by $\mu(G)$. For instance, within a path P only two robots can be placed, i.e., $\mu(P) = 2$, whereas for a ring R, $\mu(R) = 3$. In a general graph G, it turns out to be NP-complete to compute $\mu(G)$, however for a tree T, it has been proven that $\mu(T) = \ell(T)$, with $\ell(T)$ being the number of leaves of T.

After formally defining the problem of achieving GMV, starting from a configuration of robots disposed on general graphs, we focus on tree topologies. Given a tree T with $\ell(T)$ leaves, we first consider the extreme case of $n = \ell(T)$ robots disposed on $\ell(T)$ different vertices of T and we look for a distributed algorithm that makes robots moving to achieve GMV without collisions. Depending on the tree, the solution to the GMV problem is not unique, but an algorithm that moves robots to occupy all the leaves of T always solves the problem. We design a deterministic algorithm,

identical for all the robots, that solves initial configurations when considering the very weak setting of **semi-synchronous robots without lights**. For the initial configurations, where each vertex is occupied by at most one robot, we assume the absence of *critical* vertices. Intuitively, a vertex v is said to be critical if two or more equivalent robots must pass through v in order to reach a leaf, hence potentially colliding in v. The formal definition of critical vertex will be given successively. We then provide the necessary modifications to the algorithm for solving the general case with $n \leq \ell(T)$.

Furthermore, we provide an extended discussion about the configurations admitting critical vertices as well as for the asynchronous or synchronous settings. In fact, the difficulties arising in such contexts deserve deep investigation and attention. However, we provide challenging ideas, strategies and observations in order to stimulate future research.

Outline

Section 3.4.1 formalizes the GMV problem, introducing also useful notation such as the formal definition of critical vertex; Section 3.4.2 describes the proposed resolution algorithm, Section 3.4.5 discusses the complexity of the designed algorithm by providing a lower bound for GMV; Section 3.4.6 shows challenging scenarios to highlight difficulties arising when critical vertices are admitted; finally, Section 3.4.7 poses interesting future work directions.

3.4.1 Problem formulation

The topology where robots are placed on is represented by a simple and connected graph $G = (V, E)$. A function $\lambda : V \to \mathbb{N}$ gives the number of robots on each vertex of G, and we call $C = (G, \lambda)$ a *configuration* whenever $\sum_{v \in V} \lambda(v)$ is bounded and greater than zero. We introduce the Geodesic Mutual Visibility (GMV, for short) problem:

- *Given a configuration $C = (G, \lambda)$ in which each robot lies on a different vertex of a graph G, design a deterministic distributed algorithm working under the* `LCM` *model that, starting from C, brings all robots on distinct vertices – without generating collisions – in order to obtain the geodesic mutual visibility, that is there is a geodesic between any pair of robots where no other robots reside.*

As described in the introduction, we study GMV in the context of trees. In the rest of this section, we provide the necessary concepts.

GMV **on trees**

Given a configuration $C = (T, \lambda)$, a vertex v of T such that $\lambda(v) > 0$ is called *occupied*, *unoccupied* otherwise. If $\lambda(v) \geq 2$, there is a *multiplicity* in v. Given a vertex v, if v is occupied by a robot r, we often denote v also as v_r. Notation $\ell(T)$ is used to represent the number of leaves of T. As usual, $N(v)$ denotes the set of all adjacent vertices of a vertex v. Assuming $N(v) = \{v_1, v_2 \ldots, v_k\}$, the removal of v from T creates k subtrees, each denoted as $T(v, v_i)$ and assumed rooted at v_i, $i = 1, 2, \ldots, k$. If $e = (v_1, v_2)$ is an edge of T, the removal of e from T creates two subtrees, each denoted as $T(e, v_i)$ and assumed rooted at v_i, $i = 1, 2$. In each removal operation, the obtained subtrees are called *complete subtrees* of T. These removal operations are now used to provide some additional definitions.

Definition 4. *Let $C = (T, \lambda)$ be a configuration, and T' be a complete subtree of T obtained by a removal operation. T' is* overloaded *if the number of robots in T' is greater than the number of leaves of T in T'.*

Figure 3.14 shows three configurations where the complete subtrees of vertex v with robots are overloaded subtrees.

Definition 5. *Let $C = (T, \lambda)$ be a configuration. An edge $e = (v_1, v_2)$ of T is considered* oriented *from v_1 to v_2 if the complete subtree $T(e, v_1)$ is overloaded. A path $P = (v_1, v_2, \ldots, v_k)$ of T is considered* oriented *if all its edges are oriented toward the same endpoint, i.e., either v_1 or v_k. P is considered* partially-oriented *if all the oriented edges (if any) are oriented toward the same endpoint.*

Consider again Figure 3.14. All the given configurations are represented according to the edge orientation described in the above definition. In each case, the path from v_{r_1} to any unoccupied leaf is oriented, whereas the path from any occupied leaf to any unoccupied leaf is partially-oriented.

Definition 6. *Let $C = (T, \lambda)$ be a configuration, and v be a vertex of T. Vertex v is* critical *if its removal generates at least two overloaded complete subtrees $T(v, v_1)$ and $T(v, v_2)$ such that $(T(v, v_1), \lambda)$ and $(T(v, v_2), \lambda)$ are isomorphic. In such a case, these subtrees are called* critical-subtrees*. Vertex v is* potentially-critical *if its removal generates at least two overloaded complete subtrees and all such generated subtrees are pairwise non-isomorphic.*

As an example, vertex v in the configuration given in Figure 3.14.(a) is critical (in fact, $(T(v, v_{r_1}), \lambda)$ and $(T(v, v_{r_2}), \lambda)$ are isomorphic and both overloaded). In Figure 3.14.(b), instead, vertex v is potentially-critical as it is non-critical but the two sub-trees below it are both overloaded. The term potentially-critical is motivated by the observation that when moving robots from an overloaded subtree toward unoccupied leaves, the performed move could transform the vertex from potentially-critical to critical (and this, as it will be clarified in Section 3.4.6, could generate unsolvable configurations).

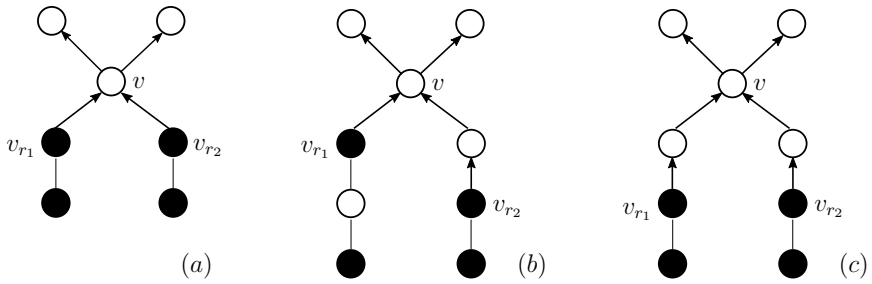


Figure 3.14: Examples of configurations. Occupied vertices are represented in black.

Definition 7. *Let $C = (T, \lambda)$ be a configuration, where $T = (V, E)$ is a tree. C is* initial *if it contains $n \leq \ell(T)$ robots, each one lying on a different vertex – that is, $\lambda(v) \leq 1$ for each $v \in V$ – and C does not contain critical vertices. C is* final *if it contains n robots each one lying on a distinct leaf of T.*

According to this definition, the specific version of GMV addressed in this work can be defined as the problem of transforming an initial configuration into a final one. Notice that a final configuration has always robots positioned on the leaves, and this ensures that GMV is solved even if the problem definition does not require such a property. In particular, given an initial configuration C, a deterministic distributed algorithm $\mathcal{A}$ *resolves* GMV if for any execution $\mathbb{E} : C = C(t_0), C(t_1), \ldots$ of $\mathcal{A}$, there exists a time instant $t^* \geq 0$ such that $C(t^*)$ is final and no robots move after t^*, i.e., $C(t) = C(t^*)$ holds for all $t \geq t^*$. Given a configuration $C = (G, \lambda)$, we say that GMV *is solvable from* C if and only if there exists a resolving algorithm for C. We will prove the following result:

Theorem 2. *Let $C = (T, \lambda)$ be an initial configuration composed of $n \leq \ell(T)$* SSync *robots, then* GMV *is solvable from C.*

Actually, we will provide the formal proof of the above theorem by considering the extreme case with $n = \ell(T)$ robots (cf., Theorem 3). Successively, we give the necessary modifications for the general case with $n \leq \ell(T)$.

In general, the solvability of many algorithmic problems defined for robots moving in a discrete or continuous environment is strongly influenced by symmetries in the input configuration, and therefore by the presence of pairwise equivalent robots. It is important to note that in this first work in which we deal with GMV, we consider symmetric configurations, but only those without critical vertices. In Section 3.4.6, we provide an extensive discussion in which we motivate how the presence of critical vertices makes solving GMV particularly difficult, if not even impossible.

3.4.2 A resolving algorithm for GMV

In this section, we provide a distributed algorithm along with its correctness proof, that is a proof for Theorem 2.

Further notation and definitions

In the following, given an initial configuration $C = (T, \lambda)$, we denote by $R = \{r_1, r_2, \ldots, r_{\ell(T)}\}$ the set of robots in C.[4]

The *center* of a graph is the set of all vertices that minimize the maximal distance from other points in the graph. It is well known that the center of a tree is a set containing one vertex or two adjacent vertices [127]. The provided algorithm requires that each robot identifies a single vertex as the center, denoted as $c(T)$. To this aim, when the center of T is a single vertex v, then each robot assumes $c(T) = v$; when the center is $\{v_1, v_2\}$ and $e = (v_1, v_2)$ is oriented toward v_i, then each robot assumes $c(T) = v_i$; when the center is $\{v_1, v_2\}$ and $e = (v_1, v_2)$ is non-oriented, each robot located in the subtree $T(e, v_i)$ assumes $T = T(e, v_i)$ and $c(T) = v_i$, $i = 1, 2$, that is like running the algorithm concurrently in two distinct instances.

In the following, given an initial configuration C, we denote by $\mathcal{P}(C)$ the set containing all the partially-oriented paths from $c(T)$ to some unoccupied leaf of T, if any. We will show that $\mathcal{P}(C) \neq \emptyset$ for each initial configuration C where GMV is not yet solved.

Definition 8. *Let $C = (T, \lambda)$ be an initial configuration. $R'(C)$ is the set containing any robot $r \in R$ such that:*

- *r is on a vertex of a path $P \in \mathcal{P}(C)$ leading to an unoccupied leaf l;*
- *r is the closest robot to l among the robots on vertices of P;*
- *the subpath of P is oriented from v_r to l.*

For each edge $e = (u, v)$ of T, let $s(e)$ be the minimum number of robots that have to pass through e to solve GMV on C. Formally, if e is not oriented, then $s(e) = 0$; if e is oriented from u to v, then $s(e)$ is the difference between the number of robots on the vertices of $T(e, u)$ and the number of leaves of T in $T(e, u)$. For a partially-oriented path P, $s(P) = \sum_{e \in P} s(e)$.

[4] We recall that we are first considering the extreme case of $n = \ell(T)$ robots and that the robots are anonymous. The notation is used only for the sake of presentation, hence no algorithm can take advantage of the names of elements in R.

The view of robots

In the algorithm, sometimes we need to distinguish among robots having some properties (e.g., minimum distance from unoccupied leaves). To this purpose, we use the so-called *view* of a robot. The view of each robot is elaborated during the `Compute` phase, starting from the snapshot perceived during the `Look` phase. In particular, we consider an approach similar to that used in [28, 108] to determine isomorphisms among trees. In particular, a robot r can associate a unique string to the tree rooted in the vertex v_r where it resides, keeping trace of the presence/absence of a robot in a vertex by associating 1 or 0, respectively. Moreover, parentheses are inserted into the strings to track the relationship between one node and its children recursively. For example, the string associated with the vertex v_{r_1} in Figure 3.15 is (1(1(1)(0))(0(1))(0)) , obtained by lexicographically ordering the strings recursively associated with the roots of its subtrees. The lexicographic order assumes "(" < ")" < "1" < "0". Since the string associated with v_{r_2} is (1(1(0(1))(0))(1)(0)), then we say that the view of robot r_1 is smaller than the view of robot r_2. Notice that two equivalent robots have the same view (i.e., are associated with the same string). In conclusion, each robot can compute the view of all robots, determine the robot(s) with minimum view, and also determine whether there is any symmetry in the observed configuration.

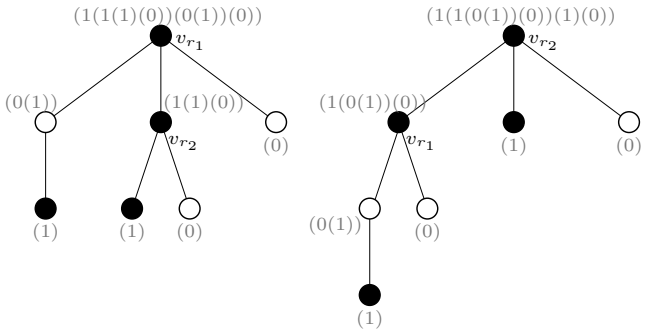


Figure 3.15: Examples of views associated with different robots/vertices.

Description of the algorithm

The provided algorithm for solving GMV from each initial configuration C is called `MoveToLeaf` and is described in Algorithm 3. Essentially, we can assume that, during the `LCM`-cycle, each robot first computes a snapshot of the current configuration (in the `Look` phase), and then executes `MoveToLeaf` (in the `Compute` phase). In the `Move` phase, the moving robot performs the move as specified in `MoveToLeaf`.

The strategy behind algorithm `MoveToLeaf` is the following. At line 2, the set $R'(C)$ is computed. If such a set is not empty, then there are robots on partially-oriented paths from $c(T)$ towards unoccupied leaves which can be brought to target by moving

Algorithm 3 `MoveToLeaf`

Input: Initial configuration $C = (T, \lambda)$.

1: compute the view of C
2: compute $R'(C)$
3: **if** $R'(C) \neq \emptyset$ **then**
4: *move* m_1*:* each robot $r \in R'(C)$ of minimum view moves toward one of its closest unoccupied leaves along a path $P \in \mathcal{P}(C)$ toward one of such leaves
5: **else**
6: let $\mathcal{S} =$ `DetermineMovingRobots`(C)
7: **if** $\mathcal{S} \neq \emptyset$ **then**
8: *move* m_2*:* let (v, v_r) be the key in $\mathcal{S}$, with r of minimum view, r moves to v

them along oriented paths. This case can be observed in Figure 3.16, where robots r_1 and r_2 belong to $R'(C)$. In this situation, the algorithm preliminarily moves these robots (cf. move m_1 at Line 4) until a configuration C_1 in which $R'(C_1) = \emptyset$ is generated.

Algorithm 4 `DetermineMovingRobots`

Input: Initial configuration $C = (T, \lambda)$.

1: let $\mathcal{S}$ be an empty map
2: compute $\mathcal{P}(C)$
3: **for all** $P \in \mathcal{P}(C)$ **do**
4: let l be the leaf where P leads
5: let v be the vertex on P closest to l such that there exists an edge $e = (u, v)$ oriented toward v with u not in P
6: **for all** occupied vertex $v_r \in T(v, u)$ **do**
7: **if** the path $P(v, v_r) = (v \equiv v_0, v_1, v_2, \ldots, v_t \equiv v_r)$ is oriented toward v **then**
8: let $\mathcal{S}[(v, v_r)] = (s((v_0, v_1)), s((v_1, v_2)), \ldots, s((v_{t-1}, v_r))$
9: let $\mathcal{S}'$ be the submap of $\mathcal{S}$ containing the lexicographically minimal sequences of $\mathcal{S}$
10: **return** $\mathcal{S}'$

Successively, since R' is empty, `MoveToLeaf` calls procedure `DetermineMovingRobots` at Line 6, the routine described in Algorithm 4. Assume that a robot r (located on some non-leaf vertex v_r) can move along an oriented path P to reach a vertex v that belongs to any partially-oriented path in $\mathcal{P}(C)$. `DetermineMovingRobots` associates a priority to r according to the integers $s(e)$ assigned to each edge e of P (an example of this assignment is shown in Figure 3.16). In this way, a sequence of integers is assigned to robots, and the robot with the lexicographically smallest sequence is then moved by `MoveToLeaf` along an oriented path toward a path in $\mathcal{P}(C)$. Notice that, if the configuration has vertices equivalent to v, equivalent robots can be selected and moved concurrently (but remember that their activation is decided by the adversary). We remark that the priority based on the integer sequences is an essential

part of the strategy as it avoids to perform "bad moves" that could transform vertices from potentially-critical to critical thus generating unsolvable configurations. For instance, the rightmost sequence represented in Figure 3.16, it can be observed that the lexicographically smallest sequence $(3, 3, 1, 1)$ is associated to the only robot that can reach v without creating critical vertices and following a path in $\mathcal{P}(C)$.

By considering again the current scenario, it follows that `MoveToLeaf` calls `DetermineMovingRobots` to select one robot (and its equivalent robots) to be moved toward a partially-oriented path from $c(T)$ to unoccupied leaves. When this path is reached by a robot, set R' turns out to be not empty and hence move m_1 is applied again to lead that robot on an unoccupied leaf. The whole process is repeated until a final configuration is created.

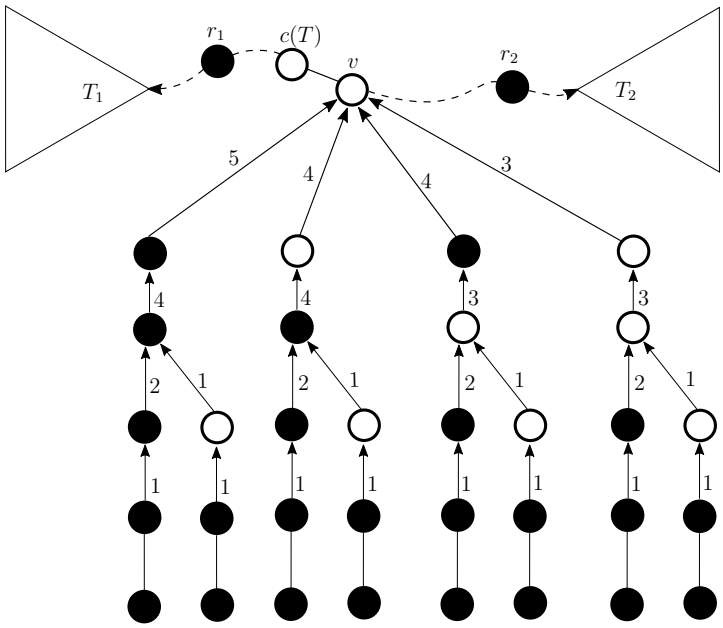


Figure 3.16: A schematic representation of an initial configuration C elaborated by `MoveToLeaf`. The triangles represent two subtrees denoted as T_1 and T_2 and containing unoccupied leaves. The dashed and curved lines represent paths. In the discussion it is assumed that the paths from $c(T)$ to T_1 and from $c(T)$ to T_2 belong to $\mathcal{P}(C)$.

3.4.3 Algorithm correctness

In this section, we prove that `MoveToLeaf` is a resolving algorithm for GMV from each initial configuration C. This is achieved by exploiting a series of lemmata concerning useful properties.

Lemma 8. *$\mathcal{P}(C) \neq \emptyset$ for each initial and non-final configuration $C = (T, \lambda)$.*

Proof. Since the configuration is non-final, there exists at least one unoccupied leaf. By contradiction, assume $\mathcal{P}(C) = \emptyset$. This implies that each path from $c(T)$ to an unoccupied leaf has at least one edge oriented toward $c(T)$. Let P_1 be one of such paths and let $e = (v_1, v_2)$ be an edge of P_1 oriented toward $c(T)$. Removing e from T generates the subtrees $T(e, v_1)$ and $T(e, v_2)$. Without loss of generality assume that $c(T)$ is contained in $T(e, v_1)$. By definition of oriented edge, in $T(e, v_1)$ the number of robots is strictly less than the number of leaves of T in $T(e, v_1)$. Hence $T(e, v_1)$ contains at least an unoccupied leaf l. Let P_2 be the path from $c(T)$ to l. Let then remove one edge oriented toward $c(T)$ from P_2 and consider again the generated subtree containing $c(T)$. Repeat this procedure until the generated subtree T^* containing $c(T)$ has no unoccupied leaf. In T^*, the number of robots is strictly less than the number of leaves of T in T^*, but the number of unoccupied leaves of T^* is zero, a contradiction. □

Lemma 9. *Let $C = (T, \lambda)$ be an initial configuration, and let C' be the configuration generated from C by* `MoveToLeaf` *according to one execution of move m_1. Then, C' is still an initial configuration.*

Proof. Consider the set R'' containing all the equivalent robots moved by move m_1. We have to show that the configuration C', created after the move of the robots in R'', is initial, i.e., it does not contain critical vertices nor multiplicities. By Lemma 8 and definition of $R'(C)$, each robot in R'' admits a distinct directed path toward an unoccupied leaf where no other robots lie. Hence, the creation of multiplicities along such paths is not possible. Let $r \in R''$ and, by contradiction, let u be a critical vertex generated after the move of r and the robots equivalent to r in R'', if any. Vertex u must be on the path from r to $c(T)$, otherwise it was a critical vertex even before the move. Since u is critical, there must be two or more pairwise isomorphic subtrees created after the move of r. Robot r must be in one of them, say $T(u, v)$. Since $T(u, v)$ is overloaded, the edge (u, v) must be oriented from v to u. This is a contradiction since (u, v) belongs to the path from $c(T)$ to the unoccupied leaf l, target of r, and this path is partially-oriented from $c(T)$ to l. □

Lemma 10. *Let $C = (T, \lambda)$ be an initial configuration in which $R'(C) = \emptyset$. Then, each $P \in \mathcal{P}(C)$ does not contain any occupied vertex.*

Proof. By contradiction, assume that there exists a path $P \in \mathcal{P}(C)$, partially-oriented from $c(T)$ to an unoccupied leaf l, with some occupied vertices. Let r be

the robot on P closest to l. Denote as P' and P'' the subpaths of P from $c(T)$ to v_r and from v_r to l, respectively. According to the definition of $R'(C)$, the assumption $R'(C) = \emptyset$ implies that P'' is not oriented toward l. Since the number of robots is equal to $\ell(T)$, there must exist an oriented path P''' from v_r to an unoccupied leaf $l' \neq l$. Since P' is partially-oriented toward v_r, then P' and P''' do not share any edge. Hence, the concatenation of P' and P''' forms a partially-oriented path from $c(T)$ to l'. A robot in this path (either r or the robot in the path that is closest to l') fulfills Definition 8. Hence $R'(C) \neq \emptyset$, against the assumption. □

Lemma 11. *Let $C = (T, \lambda)$ be an initial configuration, and let C' be the configuration generated from C by* `MoveToLeaf` *according to one execution of move m_2. Then, C' is still an initial configuration.*

Proof. Since `MoveToLeaf` applies move m_2 then $R'(C) = \emptyset$. By Lemma 10, each path $P \in \mathcal{P}(C)$ does not contain robots. Consider the set R'' containing all the equivalent robots moved by move m_2. We have to show that the configuration C', created after the move of the robots in R'', is initial, i.e., it does not contain critical vertices nor multiplicities.

Move m_2 selects a pair (v, v_r) and moves the robot r toward v. The algorithm moves all the robots equivalent to r, and then with minimal view, if any. When r is moving toward v, say from v_r to $v' \in N(v_r)$, there are two cases in which a critical vertex can be created:

1. the complete subtree $T(v', v_r)$ becomes isomorphic to another tree $T(v', a)$, hence v' becomes critical;
2. robot r becomes equivalent to a robot r'.

Case 1) Before r moves, $T(v', v_r)$ has one robot more than $T(v', a)$. Notice that there must be at least one robot in both the sub-trees. Hence, $s((v', v_r)) > s((v', a))$ and then $\mathcal{S}[(v, v_r)] > \mathcal{S}[(v, a)]$. This implies that a robot in $T(v', a)$ had to be moved instead of r.

Case 2) After the move of r on v', a new critical vertex u is created at the center of the path Q between v' and $v_{r'}$, with the two incident edges on paths $P(v', u)$ and $P(v_{r'}, u)$ oriented toward u. Moreover, u is the vertex closest to $c(T)$ among the vertices in Q. Then u is in the path $P(c(T), v)$, subpath of P, or in the path $P(v, v')$. Vertex u cannot be a vertex of $P(c(T), v)$, v excluded, due to the orientation of the edges of P toward an unoccupied leaf. Then, u is a vertex in the path $P(v, v')$ (extremes included). Since robots r and r' are equivalent, we have $s(P(v, v')) = s(P(v, v_{r'}))$. Then, $s(P(v, v_{r'}))$ is a prefix of $s(P(v, v_r))$ before the move. So, $s(P(v, v_{r'})) < s(P(v, v_r))$ and the robot to be moved was r', indeed.

As for the multiplicities, if r is the only robot moving on $v' \neq v$ then no multiplicity is possible since, by the minimality of $s(P(v, v_r))$, v' is unoccupied. If r is the only

robot moving on v' when $v' = v$, then v' is unoccupied because it is on a path $P \in \mathcal{P}(C)$ and, since $R'(C) = \emptyset$, by Lemma 10, P has no occupied vertices. If r is not the only robot moving on v', then all the robots moving on v' are equivalent, and v' is critical in C, a contradiction since C is an initial configuration. □

Theorem 3. Let $C = (T, \lambda)$ be an initial configuration composed of $n = \ell(T)$ SSYNC robots, then GMV is solvable from C.

Proof. We prove that `MoveToLeaf` is a resolving algorithm for C. Given $s(C) = \sum_{e \in E} s(e)$, it easily follows that GMV is solved from C if and only if $s(C) = 0$. Let $\mathbb{E} : C = C(0), C(1), C(2), \ldots$ be an execution of `MoveToLeaf` formed by a sequence of configurations observed at discrete time $t = 0, 1, 2, \ldots$. We have to show that there exists $t^* \geq 0$ such that $C(t^*) \in \mathbb{E}$, $s(C(t^*)) = 0$, and $C(t) = C(t^*)$ for each $t > t^*$.

Assume $s(C(0)) > 0$, and wlog $R'(C(0)) \neq \emptyset$. This implies that `MoveToLeaf` applies m_1 to $C(0)$. The resulting configuration $C(1)$ is still initial (cf. Lemma 9) and $s(C(1)) < s(C(0))$ because m_1 moves robots toward unoccupied leaves along oriented edges. Hence, by repeatedly applying m_1, `MoveToLeaf` leads to an initial configuration $C(t')$, $t' > 0$, in which $R'(C(t')) = \emptyset$. In $C(t')$, `MoveToLeaf` applies move m_2 and Lemma 11 guarantees that $C(t' + 1)$ is still initial. In particular: (1) move m_2 moves robot r (along with its equivalent robots), (2) v_r belongs to a path $P(v, v_r)$ oriented from v_r to v, and (3) r moves along $P(v, v_r)$ toward v. This implies that $s(C(t' + 1)) < s(C(t'))$. If in $C(t' + 1)$ robot r does not reach v, move m_2 is applied again. Assume that at time t'', for some $t'' > t' + 1$, r reaches v. Since v is on a path $P \in \mathcal{P}(C(t''))$, by Lemma 10 we get $R'(C(t'')) \neq \emptyset$ and move m_1 is applied again.

It follows that the execution $\mathbb{E}$ is formed by alternating subsequences of configurations in which each subsequence is generated only by move m_1 or only by move m_2. Since we have shown that the function $s()$ decreases at each execution of `MoveToLeaf`, it is clear that there exists a time $t^* > 0$ such that $s(C(t^*)) = 0$, $R(C(t^*)) = \emptyset$, and the map $\mathcal{S}$ is empty. Hence, $C(t^*)$ is final and no further moves are made. This means that `MoveToLeaf` is able to solve GMV from C. □

3.4.4 General case with $n \leq \ell(t)$ robots

So far, the proposed algorithm solves the case with $n = \ell(T)$ robots. In this section, we provide all the details necessary to deal also with $n < \ell(T)$ robots. In general, the strategy will be to add $\ell(T) - n$ virtual (and static) robots to the configuration so as to allow the use of the algorithms described before. In particular, the added virtual robots are used to compute all the directions on the edges of the input tree in order to define the set of paths $\mathcal{P}(C)$. Actually, virtual robots are not used to compute $R'(C)$ or any other subset involving robots. Of course, for consistency reasons, all robots must agree about the same locations where to add the virtual

robots. Moreover, we have to guarantee that such robots do not affect the normal functioning of the algorithms designed for the case of $n = \ell(T)$.

About the location(s) where to add $\ell(T) - n$ virtual robots, we consider the center of the input tree T. In particular, if the center of T is a set containing just one vertex $c(T)$, then robots can compute the directions of the edges by adding $\ell(T) - n$ virtual robots in $c(T)$. When the center is $\{v_1, v_2\}$, we remind that there exists the edge $e = (v_1, v_2)$. Let n_i be the number of robots residing in the subtree $T(e, v_i)$, $i = 1, 2$. Now, if v_i is not a leaf of $T(e, v_i)$ then set $\rho_i = \ell(T(e, v_i)) - n_i$, otherwise set $\rho_i = (\ell(T(e, v_i)) - 1) - n_i$. If $\rho_i > 0$, then add ρ_i virtual robots to v_i, for $i = 1, 2$. In doing so, we ensure that the role of the virtual robots never changes, so as their placement.

Consider how the proposed algorithms work. First of all, `DetermineMovingRobots` is not affected by virtual robots as by construction $c(T)$ is never part of the subtree considered by that algorithm. Concerning Algorithm `MoveToLeaf`, instead, it would move robots from $c(T)$ if the paths to the leaves do not contain robots. Since virtual robots are not accounted in $R'(C)$, the algorithm would not allow virtual robots to move. Hence, if Algorithm `MoveToLeaf` has still robots to move, it proceeds; otherwise if only virtual robots remain to move then it means GMV has been solved.

By Theorem 3 and the above discussion, we conclude the general Theorem 2 holds.

3.4.5 Time complexity

The time complexity is measured in terms of *epochs*, where an epoch is the time duration for all robots to execute at least one complete `LCM`-cycle since the end of the previous epoch. For FSync robots, an epoch coincides with a round. For SSync and Async robots, instead, the duration of an epoch may vary from time to time and it is unknown, however, by the fairness condition, it is finite.

In what follows, D denotes the diameter of the tree of a given configuration.

Lemma 12. `MoveToLeaf` *requires* $O(nD)$ *epochs to solve* GMV *from initial configurations composed of* n SSync *robots.*

Proof. The statement simply follows by observing that procedure `MoveToLeaf`, at Line 4 or at Line 8, always moves a robot at a time (along with all the robots equivalent to it) along shortest oriented paths toward a target leaf which might be of length D. □

Lemma 13. GMV *requires* $\Omega(n + D)$ *epochs to be solved from initial configurations composed of* n SSync *robots.*

Proof. Let $C = (T, \lambda)$ be an initial configuration composed of n SSync robots $r_1, r_2, \ldots, r_n$. Assume that T is a tree consisting of a path $P = (v_1, v_2, \ldots, v_n, \ldots, v_m)$

such that $m >> n$, along with $n-1$ pendant vertices attached to v_m. Assume $r_1, r_2, \ldots, r_n$ are disposed on $v_1, v_2, \ldots, v_n$, respectively. This implies that r_1 is already on a target, whereas the remaining $n-1$ robots must be moved on the $n-1$ leaves connected to v_m. In $C = C(0)$, only v_n can be moved, otherwise a collision is created. Let $C(1)$ be the configuration obtained after the move of v_n. In $C(1)$, only robots v_n and v_{n-1} can be moved. By repeating this analysis, we get that each solving algorithm can move v_2 for the first time no earlier than $t = n-2$. After this first movement, v_2 requires $D-1$ additional epochs to reach its target. At that time, GMV is solved. This implies that any solving algorithm requires $\Omega(n+D)$ epochs to solve GMV on the assumed initial configuration. □

3.4.6 On the difficulties posed by critical vertices

In this section, we illustrate some of the challenges posed by GMV on trees when critical vertices are allowed. To this aim, we show a few cases of input configurations with critical vertices that are either unsolvable or require specific strategies within SSync. Furthermore, we discuss how they can be approached within FSync.

1. Unsolvable configurations. Consider the configuration C_1 shown in Figure 3.14.(a). Vertex v is critical in C_1 since $(T(v, v_{r_1}), \lambda)$ and $(T(v, v_{r_2}), \lambda)$ are isomorphic and both overloaded (in particular, each vertex in these critical-subtrees is occupied). Notice that, in C_1 each resolving algorithm for GMV should move robots r_1 and r_2 toward v. However, since r_1 and r_2 are equivalent, no algorithm can distinguish between the two subtrees and decide which robot among r_1 and r_2 should make a step toward the parent vertex v. Hence, each algorithm would create a collision in v. Since the definition of GMV requires to not incur in collisions, then GMV results to be unsolvable in C_1, as stated in the following claim.

Claim 3.4.1. *Let C be an initial configuration in which there is a critical vertex admitting critical-subtrees having all vertices occupied. Then,* GMV *cannot be solved from C even by* FSync *robots.*

In fact, when the conditions of this claim hold, it is clear that if a robot enters (exits from or moves within) any of the critical-subtrees then a multiplicity is created.

Figure 3.14.(c) shows another unsolvable configuration in which the previous claim does not apply. Theoretically, if other robots are present, in some cases it is possible to move them inside critical-subtrees so as to break the symmetry and solve the problem. GMV cannot be solved from the configuration shown in Figure 3.14.(c) because there are no robots that can be used to break the symmetry. The following paragraph presents a deeper analysis.

2. Using leader robots to remove critical vertices. Consider the configuration shown in Figure 3.17.(a). The vertex v is critical since it has two subtrees, $T(v, v_{r_1})$ and $T(v, v_{r_2})$ that are isomorphic and overloaded. As in the previous case, it can be

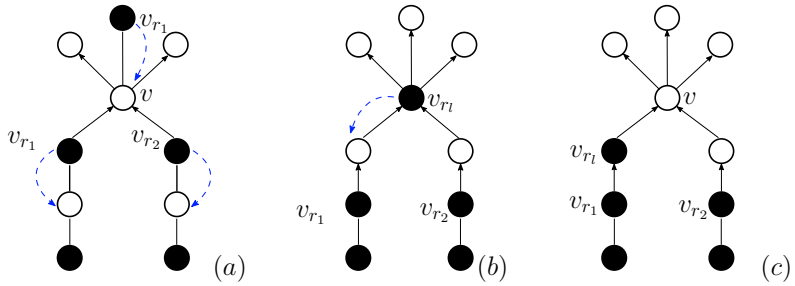


Figure 3.17: (a): A configuration with a critical vertex v. The dashed arrows show the direction proposed for the movement of the robots in order to solve GMV; (b): the symmetric configuration obtained after the one on the left with a new proposed movement; (c): the configuration made asymmetric.

observed that r_1 and r_2 cannot move directly on v otherwise they would collide. By observing that $T(v, v_{r_1})$ and $T(v, v_{r_2})$ are not completely occupied - as it happens in Figure 3.14.(a), a resolving strategy could move a robot inside one of the two isomorphic subtrees in order to break the symmetry and hence transforming v from critical to potentially-critical.

In fact, the robot r_l (which can be elected as a "leader" since it has no equivalent robots) can move toward v while r_1 and r_2 move downward. As we are in SSync (but the approach seems to be effective also in Async), such moves do not necessarily happen concurrently. In particular, if for instance r_1 moves before r_2, then the algorithm may let robots wait for r_2 to move.

After that, as in Figure 3.17.(b), r_l can move toward one of the two symmetric subtrees, hence breaking the symmetry, and thus obtaining the configuration in Figure 3.17.(c). In so doing, the critical vertex v actually becomes potentially-critical. From there, r_2 can freely move toward a leaf, and afterward r_l and r_1 in turn can reach their destination leaves, solving GMV.

Notice that the proposed strategy clearly requires (1) the presence of a leader robot outside the two isomorphic subtrees, and (2) that the involved isomorphic subtrees admit at least one unoccupied vertex where the leader robot can enter to break the symmetry. This leads to the following claim:

Claim 3.4.2. *Let C be an initial configuration in which there is a critical vertex but no leader robots. Then,* GMV *cannot be solved from C even by* FSync *robots.*

Figure 3.18.(a) shows a more general case with respect to Figure 3.17 since v has more than two isomorphic critical-subtrees. A resolution strategy moves a leader robot r_l inside one of the critical-subtrees see Figure 3.18.(a) and then the algorithm creates a configuration on other critical-subtrees so that to be different from any other configuration created on the first subtree during the emptying of the subtree. In particular in Figure 3.18.(a), r_l moves inside the first subtree making it different

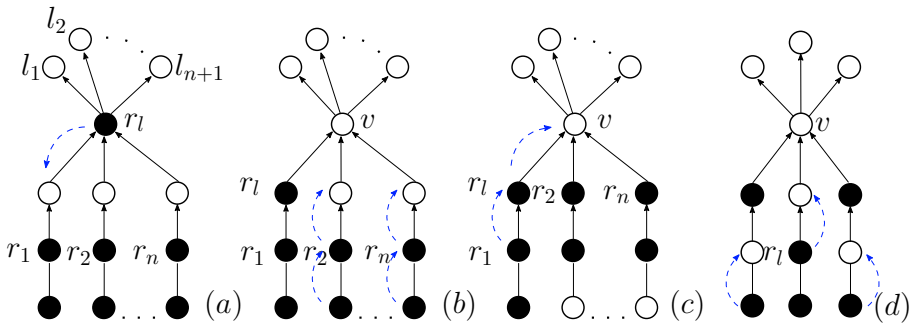


Figure 3.18: (a): A symmetric configurations with more than two isomorphic overloaded subtrees. Vertex v is critical, r_l is a leader robot, r_l moves inside the first subtree (b): robots of other subtrees, move toward v, (c): the first subtree can be emptied. (d): v is critical and r_l is the only robot that can act as leader. The robots on the leaves move up, then r_l moves toward v.

from all the other subtrees. In Figure 3.18.(b), robots r_2, ..., r_n and the ones on the leaves cautiously move one step toward v making the subtrees different from any configuration that might be created by the first subtree in the successive movements. In Figure 3.18.(c), r_l and r_1 move out of the subtree. Then all the robots that previously moved toward the root make a step back to their starting position. From here, the strategy can be repeated to move all the robots in the overloaded subtrees toward the leaves.

However a variation of the configuration of Figure 3.18.(a), shown in Figure 3.20.(a) is unsolvable in SSync. Even though a leader robot is present, it is not possible to move the robots in the other two subtrees so as to generate a configuration different from each configuration generated in the first subtree during its emptying. In fact, the robots on the leaves cannot move. Other unsolvable configurations can be generated when the leader robot cannot move, as shown in Figure 3.20.(c).

In any case, in order to solve GMV from a configuration C with a critical vertex v, it is necessary to solve the sub-problem EmptyRoots defined as follows: if v is critical, $T(v, v_1), T(v, v_2), \ldots, T(v, v_k)$ are pairwise isomorphic critical-subtrees of v, and $v_1, v_2, \ldots, v_k$ are occupied, then each resolving algorithm for GMV must be able to transform C in to a configuration C' in which $v_1, v_2, \ldots, v_k$ are all unoccupied. This is necessary so that a leader robot can move onto one of them in order to break the symmetry among the critical-subtrees. Notice that the only way to solve EmptyRoots is moving robots located in v_i downward to the corresponding critical-subtrees $T(v, v_i)$, $i = 1, \ldots, k$.

Sometimes solving EmptyRoots implies the movement of other robots (see Figure 3.19.(a)), in other cases EmptyRoots cannot be solved (see Figure 3.19.(b)). In Figure 3.19.(a), if the robot on v_{r_2} moves downward as first move, the configu-

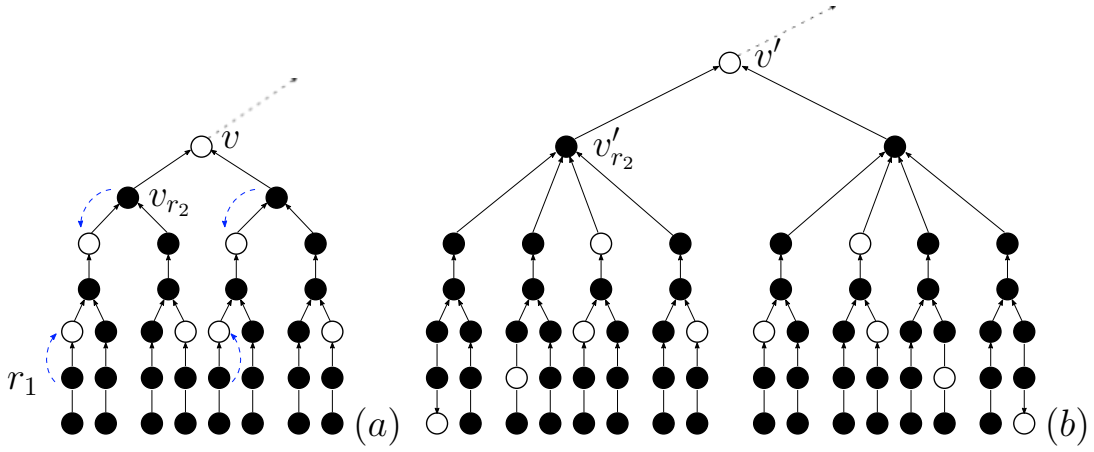


Figure 3.19: Both figures show configurations in which GMV requires solving the sub-problem EMPTYROOTS. (a) A solvable configuration where the preliminary move of r_1 upward followed by the move of r_2 downward maintains the difference between the subtrees currently rooted in v_{r_2}. (b) An unsolvable configuration.

ration becomes unsolvable; in fact, the vertex left by r_2 becomes a critical vertex being the two subtrees (below such a vertex) isomorphic and overloaded. Moreover, EMPTYROOTS must be solved for these two critical-subtrees as well, as this cannot be done without incurring in collisions. On the other hand, the configuration of the Figure 3.19.(a), can be solved by first moving robot r_1 upward and then moving r_2 downward. The movement of r_1 makes the left subtree of v_{r_2} different from the one on its right. On the contrary, the configuration shown in Figure 3.19.(b) is unsolvable because it is not possible to design a preliminary move in order to differentiate the subtrees of v'_{r_2} before moving r_2 downward.

We conjecture that deciding whether the sub-problem EMPTYROOTS can be solved could require exploring a very large number of possible moves (even an exponential number).

3. For GMV, FSYNC robots are more powerful.

Consider the configuration of Figure 3.20.(a). Since r_2 and r_3 are pairwise equivalent robots, it is not reasonable to let them both move concurrently. Instead, as shown in Figure 3.20.(b), r_l and r_1 can move concurrently toward v (we recall the reader that here we assume FSYNC robots). From there, r_l can move toward a leaf whereas r_1 can start playing the role previously played by r_l, i.e., it can move downward toward another subtree and grab another robot outside to make its role. In general, if there were $n-1$ critical-subtrees, by repeating this strategy, all the robots can be correctly moved to the leaves. Notice that this strategy cannot be implemented in SSYNC since from the configuration shown in Figure 3.20.(b) the adversary could move only

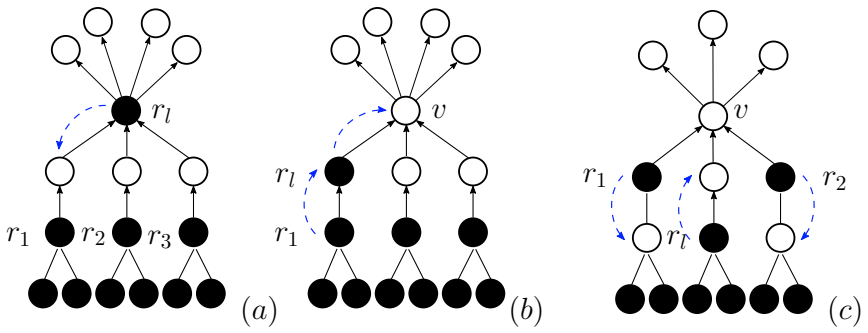


Figure 3.20: Three figures referring to symmetric configurations solvable in FSYNC but not in SSYNC.

r_l back to v creating a loop in the algorithm or only r_1, creating a multiplicity with r_l.

As an additional example, consider the configuration shown in Figure 3.20.(c). It represents a case in which there is a critical vertex v because of exactly two critical-subtrees and it is possible to elect a leader. We have already discussed a similar case in which the leader robot can move within one critical-subtree to break the symmetry and hence to solve GMV. We now show that here the problem cannot be solved in SSYNC but it is solvable in FSYNC. In fact, in SSYNC, by moving r_l or the two equivalent robots r_1 and r_2 would make the three subtrees all isomorphic. Hence, in any case v remains critical and the obtained configuration does not contain anymore leader robots. Instead, in FSYNC, the simultaneous movement of r_1, r_2 and r_l, as shown in the figure, allows to solve the problem. In fact, while r_l moves toward v, both r_1 and r_2 can move downward, and the achieved configuration becomes similar to that in Figure 3.17, where the leader robot can enter in a critical-subtree to break the symmetry. Clearly, the combined movement described cannot be applied in SSYNC.

3.4.7 Concluding Remarks

We have introduced the GEODESIC MUTUAL VISIBILITY problem for robots moving along the edges of a graph, and in particular on trees. Robots are rather weak, as they are SSYNC. The only restriction imposed to the initial configuration is the absence of critical vertices.

We have proposed a deterministic and distributed algorithm to solve GMV on trees. Furthermore, we have provided an extensive discussion about challenging directions for future research.

Chapter 4

The Moblot model

In this chapter, we introduce $\mathcal{MOBLOT}$, a new theoretical model for swarm robotics that extends $\mathcal{OBLOT}$ by introducing the concept of molecular robots. While inheriting many characteristics from the $\mathcal{OBLOT}$ model, $\mathcal{MOBLOT}$ extends it by allowing robots to cluster and create bigger computational units, called molecular robot. The goal is to model new scenarios and give new insights and algorithms for classical problems. A pattern defines the shape of a molecular robot. When robots reach relative positions that fit the pattern, from that moment on, they move together in formation, giving rise to a molecular robot. Once formed, a molecular robot is a new computational entity that moves and accomplishes new tasks, one of them can be pattern formation, at a higher hierarchical level, or it can be any other task. Three main reasons brought to the genesis of the $\mathcal{MOBLOT}$ model. The first reason is to model robots moving in formation, allowing the partition of a swarm into subgroups of robots. These subgroups can cooperate to accomplish a shared goal or can be assigned different tasks. One possible application could be molecular robots supporting humans during a fire disaster. Robots could be divided into groups and employed for various goals, like patrolling, searching, and providing radio connections for firefighters. The second reason is to introduce and exploit concepts such as modularity and hierarchy in swarm robotics under the assumptions of the $\mathcal{OBLOT}$ model in the solution of the pattern formation problem. Robots could assemble into intermediate structures (molecular robots) and then, as building blocks, molecular robots could assemble into big constructions. Such structures may have the potential flexibility to be reconfigured if goals or environmental conditions change. In this way, it would be possible to create advanced surfaces or high-tech materials. The third reason is to define a general model able also to describe modular robotics, in which robots are made of physically interconnected modular units. If fact, molecular robots can emulate modular robots by modifying the shape of their final configuration.

The inspiration for $\mathcal{MOBLOT}$ comes from chemistry, in which atoms combine to make molecules. Once bonded, molecules acquire new properties compared to single

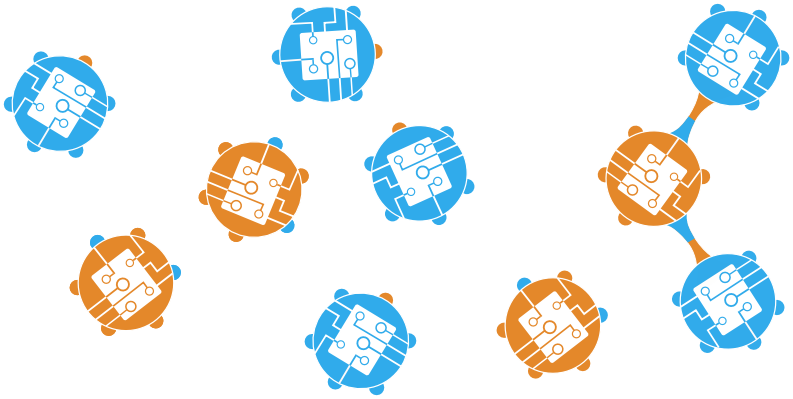


Figure 4.1: A conceptual representation of robots and a molecular robot

atoms. In many cases, molecules can further bond into more complex structures held together by intermolecular forces. One example is water, in which molecules are linked by hydrogen bonds. Molecules bond according to local composition rules, such as the angle and distance between molecules. In this way, molecular structures can scale at any dimension without defining the position of every single molecule. The molecular robots can be guided to form any shape defined according to the properties of the composition. We will use this molecular chemical paradigm to define the *matter formation* (MF) problem. MF is an extension of the Pattern Formation problem: given a team of robots, a set of formable molecules, and a geometric pattern made of molecules in an ideal coordinate system (not known to the robots nor the molecules), the goal is to provide an algorithm that guides robots to form molecules and the final pattern, if possible. Within $\mathcal{MOBLOT}$, MF is a representative problem, like PF is for $\mathcal{OBLOT}$. Hence, it is worth considering MF to investigate the new characteristics of the $\mathcal{MOBLOT}$ model. We provide the necessary conditions for the solvability of MF based on the 'amount of symmetries' of the initial configuration of robots. We show how molecules can break certain symmetries that are not solvable in $\mathcal{OBLOT}$. As an example, we consider a case study of $\mathcal{MOBLOT}$, derived from the general MF, providing a resolution distributed algorithm and proving its correctness. Moreover, we apply the $\mathcal{MOBLOT}$ model on square grids as in industrial applications, robots often are constrained to move on grid routes. To this aim, we present the Tetris-like pattern formation, an MF problem defined on grids, giving it full characterization.

Outline of the chapter. The chapter is organized as follows. The next sections introduce the $\mathcal{MOBLOT}$ model and the general Matter Formation problem. In Section 4.3, we define a case study concerning a specific matter formation problem that shows the characteristics of the $\mathcal{MOBLOT}$ model. In Section 4.3.1, we

formalize the problem and give a first overview of the resolution strategy. In Section 4.3.2, we provide details of the resolution algorithm. In Section 4.3.3, we formalize and prove the correctness of the algorithm. In Section 4.3.4, we compare the $\mathcal{OBLOT}$ and $\mathcal{MOBLOT}$ models. In Section 4.4, we apply the $\mathcal{MOBLOT}$ model to **synchronous** robots moving on a square grid; in Section 4.5, we introduce the molecular pattern formation problem (MPF for short) and introduce the concepts for its formalization, then we state a necessary condition for its feasibility. In Section 4.6 we introduce **Tetris-Like MPF** (TL-MPF for short), as a particular version of the MPF problem.

Background and related work

The robotics research is extensive, covering both computer science and engineering fields. Studies span the design, construction, operation, and use of robots. In particular, we relate to two sub-fields: *modular robotics* (e.g., see [32]) and *swarm robotics* (e.g., see [17, 125]). Modular robots are made of interconnected identical modules that allow the robot to recover from failures or change its shape to adapt to a specific task or the environment (e.g. see [141]). The main goal is to obtain robotic systems that are reconfigurable according to tasks, affordable since made up of simple modules that can be mass produced and robust. In case of failures, modules are replaceable with lower costs compared to the cost of replacing a part of an application-specific robot. However, at present, a modular robot could be less effective compared to robots designed for specific tasks. This concept was introduced in the late 1980s as cellular robotic systems by T. Fukuda, later physically realized in the CEBOT modular robot by Fukuda and Kawauchi [75]. Since then, the field is now called modular robotics, and various robotic architectures have been developed [14, 134].

On the opposite, in swarm robotics systems, robots are fully autonomous mobile units (e.g., Kilobot [123]). The interaction among robots leads to a desired collective behavior. Representative models in swarm robotics are the well-investigated Amoebot model [56, 61], and the more recent models Silbot [50, 51], and Pairbot [96]. These formal models, allow us to analyze algorithms rigorously, providing new theoretical findings, useful also in practice. Practically speaking, the technology required to implement algorithms designed within $\mathcal{OBLOT}$ does not rely on special sensors or actuators so cheap hardware can be used. An example of real robots can be found in [126]. In [118], standard educational robots under $\mathcal{OBLOT}$ solve the Gathering problem. They bring robots close to each other as much as possible. Similarly, [55] deals with the Gathering of robots moving on a ring.

Our results

We formally define the new $\mathcal{MOBLOT}$ model. $\mathcal{MOBLOT}$ extends $\mathcal{OBLOT}$ as the models coincide when a molecule is made of a single robot.

As $\mathcal{MOBLOT}$ represents an extension of $\mathcal{OBLOT}$, we introduce the Matter Formation (MF) problem as a natural extension to the classical Pattern Formation problem studied within $\mathcal{OBLOT}$. We establish a necessary condition for its solvability that relies on *symmetricity*. Molecules can resolve the symmetry-breaking issue in cases unsolvable within $\mathcal{OBLOT}$. Furthermore, we present a case study of an MF problem by specifying the formable molecules as well as their hierarchical composition rules. The considered problem later called HexMF, has been selected to highlight the symmetry-breaking abilities of the robots not present within $\mathcal{OBLOT}$. We then provide a resolution distributed algorithm for HexMF. To this respect, we show how the formal methodology thought for $\mathcal{OBLOT}$ in [44] works in the $\mathcal{MOBLOT}$ environment with molecules as well as with robots.

4.1 The model

Self-organizing structures are very common in the physical world. Atoms combine to form molecules, and they combine to form molecular structures. We use this *matter formation paradigm* to present the $\mathcal{MOBLOT}$ model.

In a $\mathcal{MOBLOT}$ system, composed by a set $R = \{r_1, r_2, \ldots, r_n\}$ of n robots, the smallest units correspond to the robots of the $\mathcal{OBLOT}$ model. As in nature there exist different types of atoms, in $\mathcal{MOBLOT}$ this can be modeled by one of the variants of $\mathcal{OBLOT}$, in which robots are colored [102]. Each robot corresponds to an atom. Colors, taken from a finite set, specify the kind of atom the robot represents.

For example, to form a pattern with the shape of the molecule of water, *white* and *red* robots correspond to hydrogen and oxygen atoms, respectively.

In a $\mathcal{MOBLOT}$ system, the algorithmic task for robots is to form **molecules**. A molecule μ is specified by a **fixed pattern** defined for the same universe $\mathcal{U}$ where the robots move. For instance, the water molecule is composed of two *white* robots and one *red* robot, where the *white* robots form a 104.5° angle with the *red* robot, and each *red-white* pair is at a distance of about 0.096 nm (cf. Figure 4.2).

Let the minimal ball enclosing a molecule μ, $B(\mu)$, and its diameter $diam(B(\mu))$. We assume $B(\mu)$ to represent the extent of molecule μ. Various shapes, different from the ball, can be defined.

We denote as $\mathcal{M} = \{\mu_1, \mu_2, \ldots, \mu_m\}$ the set containing all kinds of formable molecules. We impose some constraints for the model to be fair enough and as general/weak as possible. The first one is a **displacement constraint** for robots in any initial

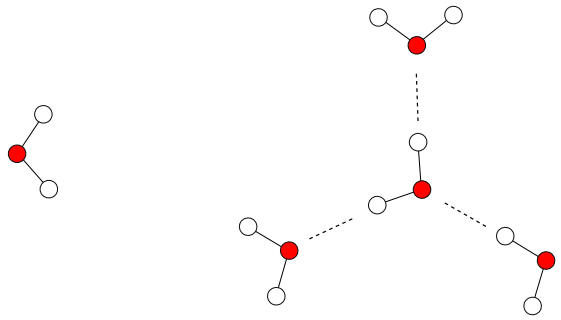


Figure 4.2: A representation of the H_2O molecule.

configuration:

$\mathcal{C}_1$: *In any initial configuration R, each pair of robots is at distance greater than $D = \max\{diam(B(\mu)) \mid \mu \in \mathcal{M}\}$.*

This constraint avoids the accidental formation of molecules in an initial configuration.

Assume that an algorithm $\mathcal{A}$ can form some molecules starting from a configuration R of robots. In the $\mathcal{MOBLOT}$ model, we assume that each robot r, performing the `Look` phase at time t, can detect the robots and formed molecules μ. r perceives both $B(\mu)$ and the robots inside the ball, that is $B(\mu)$ is "transparent". According to the ability to perceive possible formed molecules, and being the molecules expressed as fixed patterns, robots are implicitly assumed to share a common **unit of length** (for measuring distances, angles, and so on).

A molecule μ forms as soon as robots are in place for the pattern μ; there are also some additional **molecule formation constraints**:

$\mathcal{C}_2$: *In $B(\mu)$, there are only the robots necessary to form μ suitably placed according to the pattern defining μ;*

$\mathcal{C}_3$: *For each μ' already formed or that could be formed at the same time of μ, then $B(\mu) \cap B(\mu') = \emptyset$;*

$\mathcal{C}_4$: *Assume a robot r moves along a trajectory τ toward a target t and there is a position $p \neq t$ along τ such that a molecule μ forms as r in on p; if μ fulfills all constraints, it can be formed so r stops at p and the molecule forms;*

$\mathcal{C}_5$: *As soon as a molecule μ forms, each robot forming μ is no longer an independent computational unit (i.e., it stops executing its algorithm and acts as a part of the molecule).*

Remark 8. *Constraints $\mathcal{C}_2$ and $\mathcal{C}_3$ avoid ambiguities during molecule formation. Consider a molecule formed by two robots at a distance of D. As a first scenario, if three robots move synchronously to reach the vertices of an equilateral triangle with side D concurrently, then by constraints $\mathcal{C}_2$ and $\mathcal{C}_3$, none of the three possible molecules forms. Another scenario is two robots placed at a distance of $2D$ and a third one moving between them, at a distance of D from both robots. Even in this case, molecules do not form. However, as one of the two external robots moves, a molecule forms with the two stationary robots.*

Remark 9. *Robots, moving toward each other without forming any molecules, may collide. If that happens, no algorithm can guarantee to separate them anymore. As the robots are identical and occupy the same location, the adversary can prevent them from making different moves. Therefore, any algorithm must avoid collisions (i.e., undesired multiplicities) among robots.*

A molecule is a new computational entity that is solid and with a physical dimension. A molecule can move along any trajectory and also rotate around its center.[1] Since it is solid, any other element in $\mathcal{U}$ (robot or molecule) can touch the external surface of $B(\mu)$ – but cannot penetrate inside. That leads to a **movement constraint** for the model:

$\mathcal{C}_6$**:** *If the trajectory of a moving robot r intersects $B(\mu)$, for some molecule μ, then r stops its movement when it reaches the boundary of $B(\mu)$. Similarly, a moving molecule μ stops, as soon as it touches any robot r or molecule.*

The algorithm receives in input the types of molecule in $\mathcal{M}$ and the pattern of the matter defined in a general way according to **adjacency properties**. These rules define molecules' placement with respect to each other. Accordingly, the final form may vary. See Figure 4.3 for reference. Molecules may form matter by assembling in one among many possible patterns. We use the symbol $\mathcal{F}$ to denote the set containing all the possible patterns describing the matter.

4.2 The Matter Formation problem

Patter formation(PF) is one of the most general and studied problems under the $\mathcal{OBLOT}$ model.

We now recall the characterization about formable patterns in the $\mathcal{OBLOT}$ model when robots share a common chirality, hence according to the notion of symmetricity.

[1] These are basic assumptions. However, shapes or movements are changeable to suit a specific application.

Theorem 4. [131] *Let R be an initial configuration of $n \geq 3$ robots and F be a pattern. F is formable from R by* FSYNC *or* SSYNC *robots with chirality if and only if $\rho(R)$ divides $\rho(F)$.*

This result states that the solvability of the PF problem highly depends on the symmetricity of both R and F, even for FSYNC robots.

Similarly to PF, the general MATTER FORMATION (MF) problem in the $\mathcal{MOBLOT}$ model can be defined as follows: given a team of robots R, a set of molecules $\mathcal{M}$, and a set $\mathcal{F}$ of possible patterns describing the matter to form with the molecules in $\mathcal{M}$, the goal is to design a distributed algorithm $\mathcal{A}$ that works for each robot and molecule so that eventually they form any pattern in $\mathcal{F}$, if possible. Formally, $\mathcal{A}$ *solves* MF if for each execution $\mathbb{E} : R = R(0), R(1), R(2), \ldots$, there exists a time instant $t' > 0$ such that $R(t')$ is similar to some $F \in \mathcal{F}$ and $R(t) = R(t')$ for each time $t \geq t'$.

We now provide a necessary condition for the solvability of the MF problem. We assume that the spatial universe $\mathcal{U}$ in which robots and molecules move is the Euclidean plane (even though it is extendable to higher dimensions or different environments). Let R, $\mathcal{M}$, and $\mathcal{F}$ be the elements forming an instance of MF, and assume the general case in which R is composed by colored robots. A symmetry for R is an isometry φ for R (cf. Section 3.1.1) that preserves the color of robots. A symmetry for $F \in \mathcal{F}$ is an isometry for all the robots involved in F that preserves the molecules, that is if $\{r_{i_1}, r_{i_2}, \ldots, r_{i_t}\}$ is the set of robots forming a molecule $\mu \in F$, then robots in $\{\varphi(r_{i_1}), \varphi(r_{i_2}), \ldots, \varphi(r_{i_t})\}$ form a molecule equal to μ. It follows that rotational symmetries induce to the concept of symmetricity also for F. As an example, Figure 4.3 shows two patterns where the symmetricity $\rho(F) = 1$ (on the left side where also $\rho(R) = 1$) and $\rho(F) = 3$ (on the right side where $\rho(R) = 6$), respectively. We denote with $\rho(R)$ and $\rho(F)$ respectively, the symmetricity of a configuration of (possibly non-identical) robots, and the symmetricity of a matter pattern to form. We denote with $\rho(\mu)$, with $\mu \in \mathcal{M}$, the symmetricity of the set of robots that form μ. Moreover, for a given pattern $F \in \mathcal{F}$, let $Mol(F)$ denote the set of molecules that form F. Clearly, $Mol(F) \subseteq \mathcal{M}$.

We are now ready to provide a necessary condition for solving the MF problem.

Theorem 5. *Let R be an initial configuration of robots, given an instance of MF, if there exists an algorithm $\mathcal{A}$ able to form the matter, i.e., a pattern $F \in \mathcal{F}$, then*

1. *$\rho(R)$ divides $\rho(F)$, or*
2. *there exists $\mu \in Mol(F)$ such that $\rho(R)$ divides $\rho(\mu)$.*

Proof. Assume that $\mathcal{A}$ can form F without molecules (i.e., all the molecules are formed at the same time and disposed to form the matter; formally, there exists a

time $t > 0$ such that $R(t)$ is similar to F and no molecule is formed in $R(t')$ for each $t' < t$). In this case, by Theorem 4, we get that property (1) holds.

In what follows we assume that $\mathcal{A}$ must create and move some molecules to form F. We also assume $\rho(R) > 1$, otherwise, both properties (1) and (2) are trivially verified. For each possible execution $\mathbb{E} : R = R(0), R(1), \ldots$ of $\mathcal{A}$, according to Remark 1, the adversary may force $\rho(R(0))$ pairwise equivalent robots to move synchronously. Let $R(t)$, $t > 0$ be the first configuration containing molecules.

If $R(t)$ contains more than one molecule, according to the synchronous moves and to the symmetricity of R, then (i) in $R(t)$ there are $\rho(R(0))$ molecules, (ii) the molecules in $R(t)$ are all equal, and (iii) $\rho(R(t)) = \rho(R(0))$. Then, from $R(t)$ on, each move planned by $\mathcal{A}$ may be forced by the adversary to maintain at least the same symmetricity $\rho(R(0))$ until F is formed. Then $\rho(R(0))$ divides $\rho(F)$ and property (1) holds.

If $R(t)$ contains just one molecule μ, then it must be formed around the center of the configuration. Even in this case, the adversary forces $\rho(R(0))$ equivalent robots to move synchronously, and then $\rho(\mu)$ must be a multiple of $\rho(R(0))$, therefore property (2) holds. □

This theorem shows that if an algorithm $\mathcal{A}$ can form some pattern $F \in \mathcal{F}$, when the first condition does not hold, $\mathcal{A}$ must create a molecule μ in the center of the configuration and then move it to modify its symmetricity. This solution is similar to the one used in $\mathcal{OBLOT}$ to create an asymmetric configuration given a symmetric one with symmetricity 1.

Theorem 5 can be seen as the property of 'conservation' of the symmetry: the initial symmetry either is still present in the final pattern or is encapsulated in at least one molecule composing the final pattern.

4.3 A Matter Formation case study

In this section, we define a case study designed to explore all the necessary conditions of Theorem 5 and we introduce and solve a specific matter formation problem. To this end, in Section 4.3.1 we formalize the problem, motivate its study, and give a first overview of the resolution strategy. In Section 4.3.2, we provide a detailed description of the algorithm along with a running example. Finally, in Section 4.3.3, we give both a formalization and the correctness for the provided algorithm.

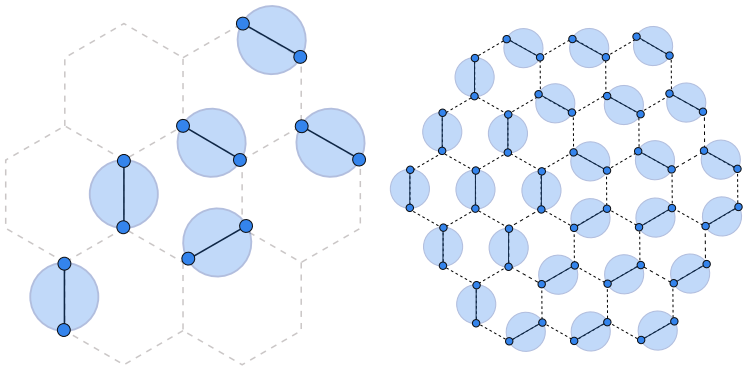


Figure 4.3: (left) matter composed of 6 molecules; (right) Matter constructed in concentric "rings". The picture shows three full levels of the matter. The inner core is formed by three molecules in the center, the second layer is composed of nine molecules all around the inner core. The external ring is composed of fifteen molecules. The hexagonal grid emphasizes the relative positions of the molecules composing the matter.

4.3.1 Problem formalization

(Hexagonal Matter Formation), (HexMF) is a specific variant of the general MF problem. In HexMF, we assume identical and ASYNC robots endowed with chirality, which is a common handedness. We remind that robots also share a common unit of measure and are aware of the quantity D necessary to form molecules. HexMF considers a single type of molecule composed of just two robots. Formally:

- $\mathcal{M} = \{\mu\}$, where μ is defined by two robots at a fixed distance D.

Remark 10. *For simplicity, in any initial configuration, the mutual distance for each pair of robots is at least $2D$. This hypothesis guarantees that a robot won't make unwanted molecules while moving in a trajectory between two points.*

In the HexMF problem, not only robots, but also molecules are assumed to be identical, ASYNC, and endowed with chirality.

Matter composition rules. We consider a hexagonal tessellation, which is a regular tiling of the Euclidean plane in which three hexagons meet at each vertex. This

tessellation induces a 'tessellation graph', which is an infinite graph embedded into the Euclidean plane induced by the vertices and sides of the hexagons forming the tessellation [89]. The tessellation graph, induced by any hexagonal tessellation, is called **hexagonal grid** and denoted as G_H. We consider the matter formed when each molecule is disposed on some edges of a hexagonal grid of side D (see Figure 4.3). According to the general definition of matter given in Section 4.1, we now specify the adjacency property for the molecules.

Consider one hexagon of G_H as the core of the matter where three non-adjacent edges of the hexagon correspond to three places where the molecules should lie. These edges form the "first level" of the matter.

Then, the six hexagons surrounding the core, represent where the "second level" of matter would be formed. That is the non-adjacent edges of the second level, not shared with the first level and parallel to those where the first three molecules are posed, correspond to the second level of edges where molecules are placed to form the matter, as soon as the first level is full.

The i-th level is formed by the non-adjacent edges of the hexagons surrounding the $(i-1)$-th level, not shared with the $(i-1)$-th level and parallel to those where the molecules of the $(i-1)$-th level are posed. Figure 4.3 (right) shows three complete levels of the matter. Actually, as in Figure 4.3.left, the last level of the matter can be not fully occupied.

It follows that $\mathcal{F}$ contains all the patterns of molecules that satisfy the above definition of matter. The matter that we defined, looks like a "polycyclic aromatic hydrocarbon", which is a chemical compound containing only carbon and hydrogen and composed of multiple aromatic rings [9].

To complete the definition of HexMF, we assume that any initial configuration (i.e., configurations considered as input for the problem) consists of a set R of robots, with $|R| = 2m$, $m > 3$. The goal is to design a distributed algorithm $\mathcal{A}$ that works for each robot and molecule so that eventually m molecules are arranged as an element $F \in \mathcal{F}$. We remark that is the responsibility of $\mathcal{A}$ to coordinate the molecules, once formed, to recognize the hexagonal grid G_H embedded into the plane. Of course, this task is difficult because both robots and molecules do not share a common reference system.

Motivations. We present the HexMF problem to show the capabilities of the $\mathcal{MOBLOT}$ model. To this end, we have defined the problem so that it constitutes a minimal and, yet well-defined, example to explore all the conditions of Theorem 5. In fact, for each specific matter F contained in $\mathcal{F}$, either $\rho(F) = 1$ or $\rho(F) = 3$. Moreover, given the unique molecule μ defined in HexMF, then $\rho(\mu) = 2$. Hence, according to the necessary conditions expressed by Theorem 5, we get that any initial configuration R from which HexMF is solvable must guarantees that the following

relationship holds:

$$1 \leq \rho(R) \leq 3. \tag{4.1}$$

According to Theorem 5, assume that there exists an algorithm $\mathcal{A}$ able to solve HexMF starting from any configuration R fulfilling the relationship in Equation 4.1. If $\rho(R) = 1$, then the first condition of Theorem 5 applies regardless of the matter F to be formed. When $\rho(R) = 2$, the second condition of Theorem 5 holds since, in this case, robots can form just a single occurrence of μ, and then this molecule could be used to "reduce" the total symmetry of the configuration to 1. Then, starting from the obtained configuration with symmetricity 1, any pattern F in $\mathcal{F}$ compatible with the size of R could be formed. Note that, in $\mathcal{OBLOT}$, if $\rho(R) = 2$ then only configurations such that $\rho(F) = 2k$ can be realized.

Finally, when $\rho(R) = 3$, the first condition of Theorem 5 holds again. In this case, Remark 1 applies, and therefore the adversary can force three robots at a time to move synchronously; of course, this necessarily leads to having three molecules formed at the beginning. It follows that a pattern F with $\rho(F) = 3$ is the only pattern that $\mathcal{A}$ could create.

Overview of the resolution strategy. Here we provide a high-level description of `FormHexMatter`, the algorithm designed for solving the HexMF problem.

Even though constraint $\mathcal{C}_1$ holds, according to Remark 10, robots can be very close to each other to create molecules correctly. To have enough space for movements, the algorithm moves almost all robots onto a circle C^* having a sufficiently large radius. To define C^*, we need a center, known to all robots. Therefore, one or three molecules (depending on $\rho(R)$) are formed around $c(R)$. The reference point for C^* is given by the "center" of the configuration of the formed molecules. By keeping these initial molecules still, the center of C^* stays fixed during the movements of the robots and all the remaining robots can reach C^*. Successively, some molecules are formed on C^*. Then, the internal molecules move to the first level of the hexagonal grid to initialize the matter; $c(C^*)$ helps define a unique embedding of the hexagonal grid G_H on the plane, becoming the center of the hexagonal grid. Later, the algorithm proceeds, by repeating the following two steps: molecules on C^* are added to the formed matter; then, new molecules are formed on C^*.

4.3.2 The resolution algorithm

In this section we detail the resolution algorithm. In particular, we firstly give some general notation and data structures, then we describe how the methodology proposed in [44] is used to break down the general problem into a set of well-defined tasks where each task can be performed by robots/molecules, and finally we provide the formalization of `FormHexMatter`.

General notation. Here we summarize general concepts and notation used by

the algorithm. Given any configuration R, the algorithm often uses circles on the plane, centered in $c(R)$. For any defined circle C, we use $r(C)$ to denote its radius. Consider any configuration R composed of robots located in distinct points of the plane, by $C_1^R, C_2^R, C_3^R, \ldots$, we denote all the circles centered on $c(R)$ such that, on each of them, is located at least one robot of R. Such circles generate a partition of R where each set of the partition contains all the robots located on the same circle. Such circles are ordered according to their radius: given two circles C_i^R and C_j^R, $i < j$ if and only if $r(C_i^R) < r(C_j^R)$. C_m is defined as the circle of radius $D/2$ centered in $c(R)$. Each half-line starting from $c(R)$ is called *ray* of $\overline{C}(R)$.

Often, n robots in R are ordered by their distance from $c(R)$. That generates a partial ordering; when we say *"the distance from $c(R)$ identifies exactly k robots"*, for any $1 \leq k < n$, we mean that such distance allows us to order the robots as follows: $r_{i_1} \leq r_{i_2} \leq \ldots \leq r_{i_k} < r_{i_{k+1}} \leq r_{i_{k+2}} \leq \ldots \leq r_{i_n}$.

We denote by Mol the set containing all the molecules formed in a configuration and by $Mat \subseteq Mol$ the subset of molecules forming the matter. According to the $\mathcal{MOBLOT}$ model, robots can directly detect Mol during the `Look` phase, whereas Mat depends on the resolution algorithm.

Robots' view. The *view of a robot* is a data structure used by the algorithm `FormHexMatter` in which each robot encodes the information acquired during the `Look` phase and refers to the configuration perceived to its LCS. Sometimes, a robot needs to evaluate the view of other robots. Therefore the view should not depend on the current LCS, as this might be completely different from cycle to cycle and from robot to robot. Therefore, the view should exploit only the information equally perceived by all robots, like relative distances and angles among robots' positions. It follows that, in a symmetric configuration, some robots are having the same view.

Given two distinct points u and v in the Euclidean plane, let $line(u, v)$ denote the straight line passing through these points and let (u, v) ($[u, v]$, respectively) denote the open (closed, respectively) segment containing all points in $line(u, v)$ that lies between u and v. The half-line starting at point u (but excluding the point u) and passing through v is denoted by $hline(u, v)$. We denote by $\sphericalangle(u, c, v)$ the angle centered in c obtained by rotating clockwise $hline(c, u)$ until overlapping $hline(c, v)$. The angle $\sphericalangle(u, c, v)$ is measured from u to v in a clockwise direction and the measurement is always positive.

Let P be a generic set of points not including $c = c(P)$. For $p \in P$, we denote by $V(p)$ the view of P computed from p. That is a sequence of pairs (angle, distance) defined as follows: first $(0, d(c, p))$ then, in order from the farthest to the closest point to c, all pairs $(0, d(c, p'))$ for any $p' \neq p$ in $hline(c, p)$, and successively all pairs $(\sphericalangle(p, c, p'), d(c, p'))$ arising from all other rays processed in clockwise order and points p' from the farthest to the closest ones to c, for each ray. Strings, associated with all points in P, can be ordered lexicographically. If $p = c(P)$ then p is said the point in P of *minimum view*, otherwise any $p = argmin\{V(p') : p' \in P\}$ is said of

problem	*sub-problem*		*task*
HexMF	*FIM*	FIM_1	T_1
		FIM_2	T_2
	MRA		T_3
	FM1		T_4
	IM	IM_1	T_5
		IM_2	T_6
	FM2		T_7
	MM		T_8
	MD		T_9

Table 4.1: The hierarchical decomposition of HexMF into tasks.

minimum view in P. These definitions naturally extend to any configuration R of robots and to any set of robots forming pattern $F \in \mathcal{F}$ as well. If r is a robot in R, by $V(r)$ we mean the view obtained from the point in which r is located. Moreover, observe that, as we are dealing with robots endowed with chirality, the clockwise direction used in the definition of the view is well-defined.

It follows that, in any symmetric configuration R, symmetric robots have the same view. Whereas, if R is asymmetric, each robot can be associated with a unique view.

Description of the algorithm. The algorithm has been designed according to the methodology proposed in [44].

Following this approach and according to the overview of the strategy described in Sec. 4.3.1, HexMF is initially divided into the following sub-problems and tasks (cf. Table 4.1):

- *Formation of the Initial Molecules*, shortly denoted as *FIM*. This task builds the first molecules defining the center of C^*. It is associated with two distinct tasks, depending on the symmetricity of the configuration. In particular, Task T_1 activates when symmetricity is either 1 or 2 (sub-problem FIM_1), while T_2 activates when symmetricity is 3 (sub-problem FIM_2).

- *Move Robots Away*, shortly denoted as *MRA*. It corresponds to task T_3, which moves all robots (not included in the initial molecules) on C^* to make enough space to construct matter in the center.

- *Forming Molecules (auxiliary case)*, shortly denoted as *FM1*. It corresponds to task T_4. It constructs some molecules on C^* before "matter initialization" by

using the initial molecules located on the center of C^*. This tasks guarantees that center of C^* remains fixed. It is considered an auxiliary task exploited to guarantee the correct evolution of the algorithm from T_3 to the subsequent tasks. The movement of molecules formed in *FIM*, without the preliminary construction of molecules on C^*, may modify the definition of C^*.

- *Initialization of the Matter*, *IM* for short. This sub-problem starts building matter by moving the molecules formed during Tasks T_1 or T_2. According to whether there are one or three molecules inside C^*, this sub-problem is subdivided into Task T_5 or Task T_6, respectively.

- *Forming Molecules*, *FM2* for short. In this sub-problem, new molecules on C^* are formed It is associated with Task T_7.

- *Moving Molecules*, *MM* for short. This sub-problem moves the molecules from C^* (through tasks T_4 or T_7) toward their target inside the circle, to construct the matter. It is associated with Task T_8.

- *Matter Done*, *MD* for short. Molecules have to detect that matter has been formed, hence no more movements are required. Task T_9 is designed for this purpose.

In the remainder of the section, we provide details for each task. In particular, we highlight the computation made by robots/molecules and formalize the outcoming moves.

Task T_1: Formation of one initial molecule

During task T_1, two robots denoted as r_1 and r_2 are identified and moved to form an initial molecule close to the center $c(R)$. Such robots move differently according whether $c(R)$ is occupied or not. The resulting move is denoted as m_1 and formalized as follows.

- **Move** m_1:
  - If there is a robot r on $c(R)$, then r radially moves toward C_m in any direction that does not meet any robot (we recall that C_m is the circle of radius $D/2$ centered in $c(R)$).
  - If $c(R)$ is not occupied, two robots r_1 and r_2 are selected and moved to form an initial molecule.

    Let r_1 be the robot closest to $c(R)$ (of minimum view in case of ties), ℓ_1 be the line passing through r_1 and $c(R)$, ℓ_2 be the line orthogonal to ℓ_1 and passing through $c(R)$, $\mathcal{H}_1$ and $\mathcal{H}_2$ be the half-planes defined by ℓ_2 with r_1 contained in $\mathcal{H}_1$. Associate to each robot a pair (d_1, d_2), where d_1

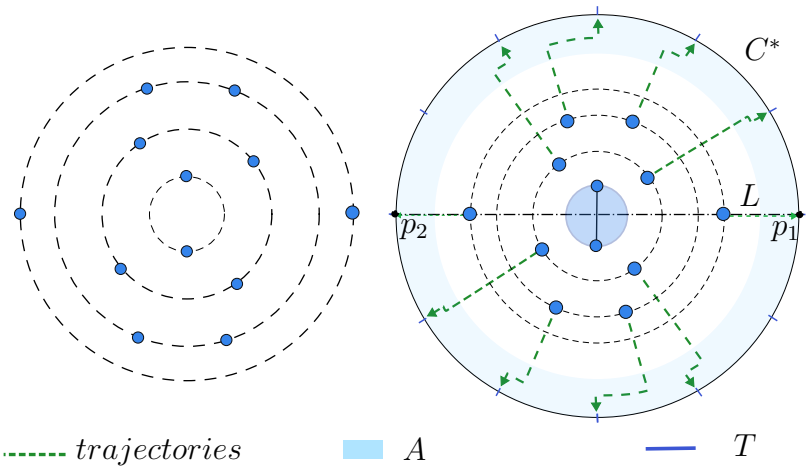


Figure 4.4: Configurations belonging to tasks T_1 (left) and T_3 (right). Since R contains 12 robots, in the configuration belonging to T_3 the radius of C^* should be of at least $12D$; in the figure the relative distances are not respected in favor of readability.

is the distance from $c(R)$ and d_2 is the distance from ℓ_1, and consider such pairs as lexicographically ordered. Now, consider the set S containing all robots different from r_1 and with minimum associated pair (d_1, d_2). If S has robots in $\mathcal{H}_2$, then select r_2 as the robot in $\mathcal{H}_2$ with minimum view, otherwise r_2 is the robot in $\mathcal{H}_1$ with minimum view.

The selected robots r_1 and r_2 move as follows: if r_1 is not on C_m, it moves radially on C_m, when r_1 is on C_m, r_2 moves radially toward C_m. Since r_1 and r_2 reduce their distance while moving toward C_m eventually a molecule with robots r_1 and r_2 is formed. The distance $2D$ between each robot in any initial configuration guarantees there is no chance to form any other molecule except the one between r_1 and r_2 while they move.

According to the definition of r_1 and r_2, when the symmetricity of the initial configuration is 2, then r_1 and r_2 result to be antipodal (as in Figure 4.4.left). In this case, the two robots are equivalent and they move concurrently toward each other (it can be observed that the molecule is finally formed even in case of asynchronous moves).

Task T_2: Formation of three initial molecules

During task T_2, three robots denoted as r_1, r_2, and r_3 are identified and moved toward the other three robots denoted as r'_1, r'_2, and r'_3 to form three initial and distinct molecules. The moving robots are selected among the most internal robots of $\overline{C}(R)$, so the resulting molecules will be close to the center $c(R)$.

Two perform this task on an initial configuration R, robots must recognize that

$\rho(R) = 3$. Unfortunately, due to possible asynchronous movements, before the requested molecules are formed, some intermediate configurations with symmetricity different from three can be created and hence observed by some robots. This observation imposes to provide robots with some *pre-conditions* that allows them to correctly recognize not only the initial but also each possible intermediate configuration created during the task execution.

Since the algorithm considers two different strategies for creating the molecules, two different Boolean pre-conditions are defined.

`iM3′`: It is considered true when all the following properties hold (cf. Figure 4.5):

1. the distance from $c(R)$ identifies exactly r_1, r_2, r_3;
2. if $R' = R \setminus \{r_1, r_2, r_3\}$, then $\rho(R') = 3x$, $x > 0$;
3. at least one robot among r_1, r_2, r_3 is not part of a molecule;
4. the rays passing through r_1, r_2, r_3, respectively, and rotating clockwise, meet three robots r'_1, r'_2, r'_3 on $C_1^{R'}$ at 120^o each other;
5. r_1, r_2, and r_3 are on the same circle *or* their projections on $C_1^{R'}$ coincide with r'_1, r'_2, r'_3, respectively.

`iM3″`: It is considered true when all the following properties hold (cf. Figure 4.6):

1. C_1^R contains more than three robots;
2. there exist r_1, r_2, r_3 on C_1^R such that: their distance to the next (clockwise) robots is minimum and their rotation toward the next (clockwise) robots generates a configuration R' with $\rho(R') = 3$;
3. at least one robot among r_1, r_2, r_3 is not part of a molecule.

The resulting move is denoted as m_2 and formalized as follows.

- **Move** m_2:
  - If `iM3′` holds, robots r_1, r_2, and r_3 first rotate clockwise along C_1^R without ever reaching distance D from the next robot, until they are all aligned with r'_1, r'_2 and r'_3 lying on $C_1^{R'}$. If they reach such an alignment without creating molecules, then they move radially toward r'_1, r'_2 and r'_3, respectively, until forming three molecules.
  - If `iM3″` holds, then robots r_1, r_2, and r_3 rotate clockwise along the circle, until three molecules are formed.

Note that when applying the move, the same pre-condition between `iM3′` and `iM3″` that was true at the beginning of the task remains valid until the task is completed.

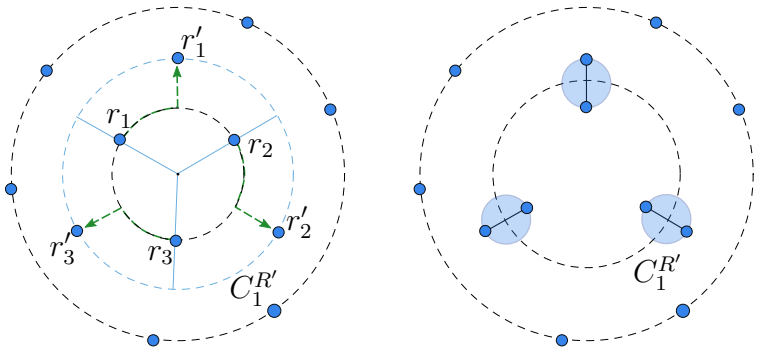


Figure 4.5: (left) configuration in T_2 where pre-condition `iM3'` holds; (right) configuration obtained at the end of T_2 and belonging to T_3.

Task T_3: Moving robots away

The only molecule formed in T_1 or the three molecules formed in T_2 are used in this task to detect a center from which C^* is identified. Let R be a configuration obtained after task T_1 or T_2. Circle C^* is defined as follows:

- In case only one molecule μ is formed in R, then C^* is the circle with the same center as the ball $B(\mu)$, including all the robots, with radius given by the integer

  $$\rho_1 = \min\{d \mid d \geq 2mD \ \wedge \ d \textit{ is a multiple of } 2m\}$$

  admitting an annulus A delimited by C^* and by a circle of radius $r(C^*) - 3D$ where at most one robot resides.

- In case three molecules are formed and are included in a minimum enclosing circle C of radius x, then C^* is the circle with the same center as C, including all the robots, with radius given by the integer

  $$\rho_3 = \min\{d \mid d \geq 2mD + x \ \wedge \ d \textit{ is a multiple of } 2m\}$$

  admitting an annulus A delimited by C^* and by a circle of radius $r(C^*) - 3D$ where at most three robots reside, one for each sector of C^* (the partition into sectors of the area enclosed by C^* is defined later in this section, just before the formal definition of move m_3).

The annulus used in this definition can be observed in Figure 4.4.right.

The aim of this task is to move all the robots not forming molecules so that the following property holds:

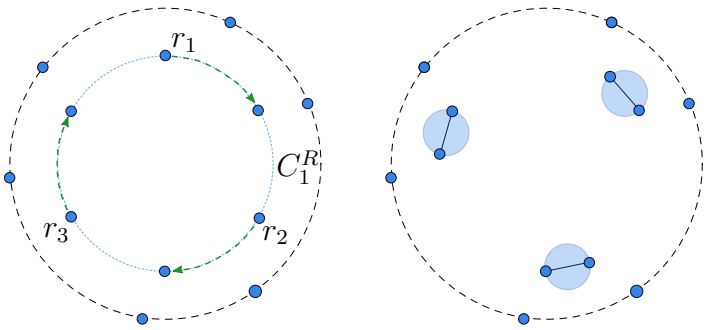


Figure 4.6: (left) configuration in T_2 where pre-condition iM3″ holds; (right) the configuration obtained at the end of T_2 and belonging to T_3.

FarC: All robots (excluding those forming molecules) are correctly positioned on C^*.

Task T_3 is characterized by the following pre-condition.

cM: It is considered true when all the following properties hold:

1. ¬iM3′
2. ¬iM3″
3. $|Mol| = 1$ *or* $(|Mol| = 3$ *and* $\rho(Mol) = 3)$
4. ¬FarC

In particular, Properties cM. 1 and cM. 2 ensure that the initial molecules have been already formed, Property cM. 3 assures that the formed molecules have the right symmetricity, and Property cM. 4 states that the task is not yet completed.

to move the robots on C^*, we need to define a suitable set T of target points on C^*. Figure 4.4.right provides an example of the following concepts and notation necessary to define T. When $|Mol| = 1$, let L be the line passing through $c(C^*)$ and orthogonal to the segment between the two robots forming the molecule, and let $P = \{p_1, p_2\}$ with p_1 and p_2 being the intersections of L with C^*. When $|Mol| = 3$ instead, let L_1, L_2 and L_3 be the radii of C^* passing through the center of each molecule, then $P = \{p_1, p_2, p_3\}$ with p_1, p_2 and p_3 being the intersections of L_1, L_2 and L_3, respectively, with C^*. The set T is defined by all the points at a distance multiple of $\pi r(C^*)/m$ from points in P in the clockwise direction on C^*. Being C^* of radius multiple of $2m$, the points of T are $2m$, including those in P, and are equally distributed on C^*.

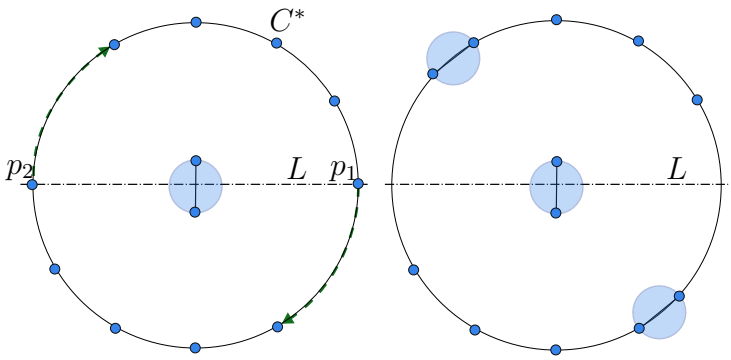


Figure 4.7: Configurations in tasks T_4 (left) and T_5 (right). Note that in T_4, the robots that move are those on p_1 and p_2.

The area enclosed by C^* can be partitioned into $\rho(Mol)$ sectors as follows: when $\rho(Mol) = 3$ the sectors are defined by the radii L_1, L_2 and L_3; when $\rho(Mol) = 2$ they are defined by the half-lines forming L; finally, when $\rho(Mol) = 1$, the whole area inside C^* forms just one sector.

The resulting move to lead all the robots on the target in T is denoted as m_3 and formalized as follows.

- **Move** m_3:
  - Robots are moved on C^* so as to not create undesired molecules. For each sector, and in a coordinated 3-steps way, the robot furthest from $c(C^*)$ is first moved radially until distance $\frac{3}{2}D$ from C^* (that is in the exact middle of A), then it rotates clockwise until being on the radius of C^* passing through the *first unoccupied target*, and finally moves radially to the target.

Note that there might be at most three robots moving concurrently. The use of annulus A is to be sure that the moving robots do not create molecules accidentally while moving. In fact, the width of A is $3D$ and robots move in the middle of A, that is at a distance of at least $1.5D$ from any other robot.

Task T_4: Forming molecules - auxiliary case

The task is needed when T_3 is finished and the matter can be formed (through the subsequent tasks). In particular, T_4 can be thought as an auxiliary task exploited to guarantee the evolution of the system from T_3 to T_5 or T_6. In fact, the formation of the matter without creating the molecules handled by T_4 may result in a modification of the definition of C^*.

We now introduce some additional notation. Let Mol' be the set of molecules inside C^*. We recall that in the description of task T_3 there are defined points p_1 and p_2 on C^* when $|Mol'| = 1$, whereas such points became p_1, p_2, and p_3 when $|Mol'| = 3$. Now, if $|Mol'| = 1$, let $X = \{r_1, r_2\}$ be the first two robots that are met from p_1 and p_2, respectively, in the clockwise direction. Conversely, if $|Mol'| = 3$, let $X = \{r_1, r_2, r_3\}$ be the first three robots that are met from p_1, p_2, and p_3, respectively, in the clockwise direction. Once X is defined, let $R^+ = R \setminus X$. Figure 4.7.left shows a configuration processed by Task T_4

This notation can now be used to define the pre-condition that characterizes task T_4.

`nM1:` It is considered true when all the following properties hold:

1. $|Mat| = 0$
2. $|Mol'| = 1$ *or* $|Mol'| = 3$
3. `FarC`
4. the number of molecules on C^* is less than $\rho(R^+)$

The move performed in this task is denoted as m_4 and formalized as follows.

- **Move** m_4:
  - If $|X| = 3$ or ($|X| = 2$ and $\rho(R^+) = 2$), then all robots in X rotate clockwise, otherwise among r_1 and r_2 the farthest from L rotates clockwise.

According to this move, one, two or three molecules are formed on C^* depending on the possible initial symmetry deduced from $\rho(R^+)$. These molecules, along with the positioning of the other robots on C^*, allow the movement of the internal molecules to create the core of the matter made by either one or three molecules. Figure 4.7.right shows a configuration obtained at the end of Task T_4 when $\rho(R) = 2$.

Tasks T_5: One molecule initializes the matter

This task is devoted to "initializing the matter" by correctly positioning the unique molecule formed in task T_1. Concretely, an embedding of the hexagonal grid G_H into the plane is defined, and the unique molecule located close to the center of C^* is moved on an edge of the first level of G_H. In previous tasks, C^* was identified by robots by using the internal molecules formed during T_1 or T_2. Now, after moving the internal molecule, C^* is recognized as *the circle containing all the robots not forming molecules and two robots forming one molecule and with radius equal to* ρ_1 (for ρ_1 see Section 4.3.2). Given this definition, internal molecules can freely move without changing the identification of C^*.

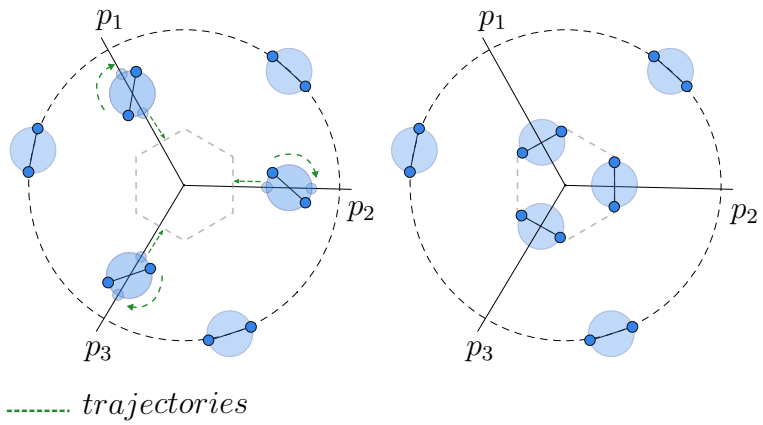


Figure 4.8: Movement of molecules in Task T_6.

A preliminary embedding of G_H is given by matching its center with the center of C^*. The exact embedding will be defined once the molecule μ is moved to a position consistent with the first level of G_H. Figure 4.7.right provides an example of a configuration where this task must be applied.

This task is applied only when the following pre-condition holds.

`nM2:` It is considered true when all the following properties hold:

1. $|Mat| = 0$
2. `FarC`
3. $2 \leq |Mol| \leq 3$: 1 or 2 molecules are on C^* and 1 internal
4. the number of molecules on C^* no less than $\rho(R)$

All these properties remain valid during the movement of μ, while the first one becomes false as soon as μ reaches the target. The move performed in this task is denoted as m_5 and formalized as follows.

- **Move** m_5:
  - The unique internal molecule μ radially moves along L until reaching a position consistent with the first level of the defined embedding of G_H.

Note that if the initial configuration was admitting symmetricity equal to 2, after T_5 the configuration becomes asymmetric. This symmetry breaking is impossible in the $\mathcal{OBLOT}$ model.

Tasks T_6: Three molecule initialize the matter

This task is similar to T_5, but here three molecules instead of one are used to "initialize the matter". As in the previous task, C^* is recognized as *the circle containing all the robots not forming molecules along with six robots forming three molecules, and with radius equal to* ρ_3 (for ρ_3 see Section 4.3.2). Given this definition, internal molecules can freely move without changing the identification of C^*. Again G_H is embedded by matching its center with the center of C^*. The exact embedding will be defined once the molecules are located close to the center of G_H in a position consistent with the first level of G_H. Figure 4.8.left provides an example of a configuration where this task must be applied.

This task is applied only when the following pre-condition holds.

`nM3:` It is considered true when all the following properties hold:

1. $0 \leq |Mat| < 3$
2. `FarC`
3. $|Mol| = 6$: 3 molecules are on C^* and 3 internal

All these properties remain valid during the movement of the three internal molecules, while the first one becomes false as soon all the moving molecules reach the target. The move performed in this task is denoted as m_6 and formalized as follows.

- **Move** m_6:
  - The three internal molecules first rotate clockwise with respect to their center until the rays of C^* passing through their centers become orthogonal to the segments joining the two robots forming each molecule. Then, they radially move until reaching the right positioning to become part of the matter with respect to $c(C^*)$.

Once this task ends, the matter is suitably initialized and the configuration admits a symmetricity of three.

Task T_7: Forming new molecules

This task forms new molecules on C^* that later will be moved to grow the matter. It starts when the matter is composed of at least three molecules and no new molecules on C^* exist. The difficulty here is mainly due to detecting how many molecules on C^* need to be formed and selecting and moving the robots to form new molecules on C^*.

For this aim, we refer again to the lines L_1, L_2, and L_3 introduced in the description of task T_3 (cf. Section 4.3.2), and to the points p_1, p_2, and p_3 induced by such lines

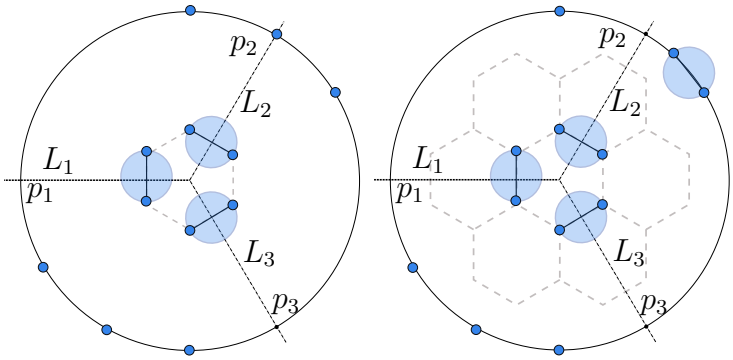


Figure 4.9: Configurations in tasks T_7 (left) and T_8 (right).

on C^*. In the move planned for this task, the algorithm selects the robots r_1, r_2 and r_3 as the first three robots met from p_1, p_2, and p_3, respectively, in the clockwise direction.

The move performed in this task is denoted as m_7 and formalized as follows.

- **Move** m_7:
  - If the matter is currently composed by at least three molecules and the first three robots r_1, r_2, r_3 met from p_1, p_2, and p_3, respectively, in the clockwise direction, are distinct and $\rho(R \setminus \{r1, r2, r3\}) = 3$, then r_1, r_2 and r_3 rotate along C^* until creating three molecules. In any other case, the robot on C^* closest to the successive one in the clockwise direction, with minimum view in case ot ties, rotates along C^* until forming a new molecule.

This move builds either one or three molecules on C^*. We conclude by observing that this task is applied only when the following pre-condition holds.

`M1:` It is considered true when all the following properties hold:

1. `FarC`
2. $|Mat| > 0$
3. $|Mol \setminus Mat| < 3$

All these properties remain valid during the movement of the robots in C^*, while the third one becomes false if three molecules are formed on C^*. In Section 4.3.3, we will see that even though `M1` may remain true once T_7 is over, the subsequent Task T_8 has a higher priority, so it will be responsible to correctly detect such an occurrence (see, e.g. Figures 4.9 and 4.10).

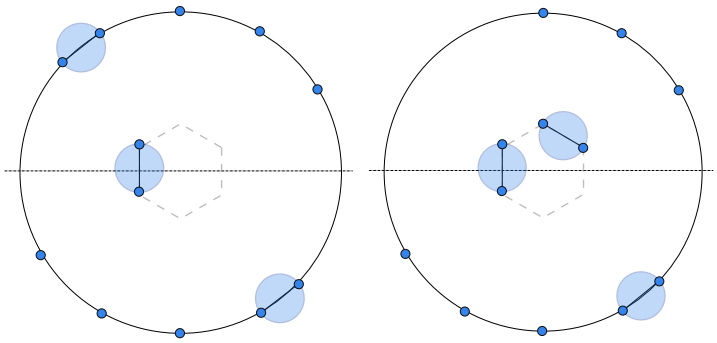


Figure 4.10: Two successive configurations belonging to task T_8.

Task T_8: Moving molecules

This task moves all the molecules – previously formed on C^* through tasks T_4 or T_7 – to grow the matter. As in the previous task, here we still refer to the three lines L_1, L_2, and L_3 introduced in the description of task T_3, and to the points p_1, p_2, and p_3 induced by such lines on C^*. Hence, robots r_1, r_2 and r_3 are defined as the first three robots on C^* met from p_1, p_2, and p_3, respectively, in the clockwise direction.

Here the algorithm exploits an extension of the notion of symmetricity, since this notion is also applied to the matter formed so far. In particular, by $\rho(Mat)$ we denote the symmetricity of the molecules already embedded in the hexagonal grid G_H. For instance, in Figure 4.10 $\rho(Mat) = 1$, in Figure 4.9 $\rho(Mat) = 3$, in Figure 4.3 $\rho(Mat) = 1$ and $\rho(Mat) = 3$ on the left and right side, respectively.

T_8 is applied when the following pre-condition holds in the observed configuration.

`M2`: It is considered true when both the following properties 1 and 2 hold:

1. $|Mat| > 0$
2. The following does not hold: there are three molecules on C^* and $|Mol' \setminus Mat| > 0$.
3. One of the following properties holds:
   (a) $|Mol \setminus Mat| > 0$ *and* $\rho(Mat) = 1$
   (b) $|Mol \setminus Mat| = 1$ *and* $\rho(Mat) = 3$ *and* $\rho(R \setminus \{r_1, r_2, r_3\}) = 1$
   (c) $|Mol \setminus Mat| = 3$.

Concerning Property 1, it is required to be sure that the matter has been previously correctly initialized. Moreover, it stays true during the whole task since it is not affected by the move (the matter can only increase).

Concerning Property 2, it is responsible for recognizing whether T_6 is over (i.e., all the three molecules inside C^* have been moved to correctly initialize the matter)

or not. In fact, as it will be clarified in Section 4.3.3, since the precondition of T_8 is evaluated before that of T_6, here it is necessary to test whether there are three molecules on C^* and, at the same time, there are molecules internal to C^* (i.e., molecules in Mol') which are not yet correctly added to the current matter. If this condition holds, then it is clear that T_6 is not completed.

Let us now consider Property 3. This is due to the fact that, similarly to T_6, in the algorithm this pre-condition is evaluated before that of T_7 and hence it is responsible of recognizing whether T_7 is over (i.e., no more molecules must be formed on C^*) or not. It is based on three different sub-properties:

- Property 3a captures configurations where there are molecules on C^* and $\rho(Mat) = 1$. If T_8 is just starting, then there are one or two molecules on C^*, no more molecules must be formed on C^* and T_8 can proceed; otherwise, $\rho(Mat) = 1$ could be due to some molecule just added and also in this case it is correct to start again T_8 (it may happen that there are other two molecules to be moved).
- Property 3b captures configurations where there is one molecule on C^* and $\rho(Mat) = 3$. Condition $\rho(R \setminus \{r_1, r_2, r_3\}) = 1$ allows the algorithm to recognize that the current configuration must not be processed by T_7 (no more molecules to be created on C^*). Hence, the unique molecule on C^* must be added to the matter.
- Property 3c is simple: here there are three molecules on C^*, and since T_7 builds at most three molecules, it is evident that T_8 must be started.

The move performed in this task is denoted as m_8 and formalized as follows.

- **Move** m_8:
  - Each molecule moves toward the closest available position of the last level of the matter not yet filled, in the clockwise direction in case of ties, while possibly rotating with respect to its center (e.g., see Figure 4.10).

Task T_9: Matter done

It refers to the requirement of letting molecules to detect the matter has been formed, hence no more movements are required. The corresponding precondition is defined as follows:

`Mf:` All robots form molecules *and* matter is completed

Clearly, only *nil* movements are allowed and it is not possible to switch to any other task (e.g., see Figure 4.3).

4.3.3 Algorithm formalization and correctness

`FormHexMatter` is the algorithm designed to solve the HexMF problem. It is based on a strategy that decomposes HexMF into tasks $T_1, T_2, \ldots, T_9$. Tasks have been detailed throughout Section 4.3.2. For each task we have provided a detailed description for (1) the concept and notation specifically needed by the task, (2) the pre-condition that must be verified to accomplish the task, and (3) the move performed by robots/molecules for accomplishing the task. For the sake of convenience, Table 4.2 summarizes all the Boolean variables introduced to define the tasks' preconditions.

According to the algorithm design methodology introduced in [44] and recalled in Section 3.2, we state the following lemma.

Lemma 14. *Predicates P_i fulfill both Properties* Prop_2 *and* Prop_3.

Proof. Prop_2 is directly implied by Equation 3.1. Prop_3 is implied by pre-condition $\mathtt{pre}_1$ and predicates P_i. □

Table 4.3 formalizes the proposed algorithm: the first two (general) columns recall the hierarchical decomposition, the third column associates tasks names to subproblems, and the fourth column defines precondition $\mathtt{pre}_i$ for each task T_i. These preconditions must be used to define each predicate P_i according to Equation 3.1. The fifth column of Table 4.3 contains the name of the move defined for each task. The last column specifies the **transitions** $T_i \to T_j$ that can occur from each task T_i, that is any possible task T_j to be performed on the obtained configurations once T_i is terminated. For example, the table states that from T_4 only transitions $T_4 \to T_5$ and $T_4 \to T_6$ can occur.

According to the definitions of P_i given in Equation 3.1, in the `Compute` phase, each robot evaluates – with respect to the perceived configuration and the provided input μ and $\mathcal{F}$ – the preconditions starting from P_9 and proceeding in the reverse order until a true precondition is found. In case all predicates $P_9, P_8, \ldots, P_2$ are evaluated false, then task T_1, whose precondition is simply `true`, is performed.

Predicates evaluation: an example. In this paragraph, we provide an example of how robots, during the execution of `FormHexMatter`, detect the task to be performed. For this purpose, we consider distinct initial configurations, those shown in Figures 4.4.left, 4.5 and 4.6.

The initial configuration in Figure 4.4.left is such that $\rho(R) = 2$ and belongs to T_1. In fact, since there are no molecules formed, `Mf`, `M2`, `M1`, `nM3`, `nM2`, `nM1`, and `cM` are false, that is the configuration is not in $T_9, \ldots T_3$, respectively. Concerning `iM3''`, circle C_1^R contains only two robots, hence the predicate is false. Concerning `iM3'`, the distance from $c(R)$ does not identify exactly three robots, hence the predicate is false too, that is the configuration is not in T_2 and then belongs to T_1. During

prec.	*var*	*definition*
$*$	`FarC`	True when all robots (excluding those forming molecules) are correctly positioned on C^*.
$\texttt{pre}_2$	$\texttt{iM3}'$	True when all the following properties hold: 1. the distance from $c(R)$ identifies exactly r_1, r_2, r_3; 2. if $R' = R \setminus \{r_1, r_2, r_3\}$, then $\rho(R') = 3x$, $x > 0$; 3. at least one robot among r_1, r_2, r_3 is not part of a molecule; 4. the rays passing through r_1, r_2, r_3, respectively, and rotating clockwise, meet three robots r'_1, r'_2, r'_3 on $C_1^{R'}$ at 120^o each other; 5. r_1, r_2, and r_3 are on the same circle *or* their projections on $C_1^{R'}$ coincide with r'_1, r'_2, r'_3, respectively.
	$\texttt{iM3}''$	True when all the following properties hold: 1. C_1^R contains more than three robots; 2. there exist r_1, r_2, r_3 on C_1^R such that: their distance to the next (clockwise) robots is minimum and their rotation toward the next (clockwise) robots generates a configuration R' with $\rho(R') = 3$; 3. at least one robot among r_1, r_2, r_3 is not part of a molecule.
$\texttt{pre}_3$	`cM`	True when all the following properties hold: 1. $\neg\texttt{iM3}'$ 2. $\neg\texttt{iM3}''$ 3. $\lvert Mol\rvert = 1$ *or* $(\lvert Mol\rvert = 3$ *and* $\rho(Mol) = 3)$ 4. $\neg$`FarC`
$\texttt{pre}_4$	`nM1`	True when all the following properties hold: 1. $\lvert Mat\rvert = 0$ 2. $\lvert Mol'\rvert = 1$ *or* $\lvert Mol'\rvert = 3$ 3. `FarC` 4. the number of molecules on C^* is less than $\rho(R^+)$
$\texttt{pre}_5$	`nM2`	True when all the following properties hold: 1. $\lvert Mat\rvert = 0$ 2. `FarC` 3. $2 \leq \lvert Mol\rvert \leq 3$: 1 or 2 molecules are on C^* and 1 internal 4. the number of molecules on C^* is no less than $\rho(R)$
$\texttt{pre}_6$	`nM3`	True when all the following properties hold: 1. $0 \leq \lvert Mat\rvert < 3$ 2. `FarC` 3. $\lvert Mol\rvert = 6$: 3 molecules are on C^* and 3 internal
$\texttt{pre}_7$	`M1`	True when all the following properties hold: 1. `FarC` 2. $\lvert Mat\rvert > 0$ 3. $\lvert Mol \setminus Mat\rvert < 3$
$\texttt{pre}_8$	`M2`	True when all the following properties hold: 1. $\lvert Mat\rvert > 0$ 2. The following does not hold: there are three molecules on C^* and $\lvert Mol' \setminus Mat\rvert > 0$ 3. One of the following properties holds: (a) $\lvert Mol \setminus Mat\rvert > 0$ *and* $\rho(Mat) = 1$ (b) $\lvert Mol \setminus Mat\rvert = 1$ *and* $\rho(Mat) = 3$ *and* $\rho(R \setminus \{r_1, r_2, r_3\}) = 1$ (c) $\lvert Mol \setminus Mat\rvert = 3$
$\texttt{pre}_9$	`Mf`	Matter formed.

Table 4.2: Summary of all the Boolean variables defined in the description of tasks $T_1, T_2, \ldots, T_9$ (cf. Section 4.3.2), used to define the tasks' preconditions $\texttt{pre}_2, \ldots, \texttt{pre}_9$.

problem	*sub-problem*		*task*	*precondition*	*move*	*transitions*
HexMF	*FIM*	FIM_1	T_1	*true*	m_1	T_3, T_4
		FIM_2	T_2	$\texttt{iM3}' \vee \texttt{iM3}''$	m_2	T_3, T_4
	MRA		T_3	`cM`	m_3	T_4
	FM1		T_4	`nM1`	m_4	T_5, T_6
	IM	IM_1	T_5	`nM2`	m_5	T_8
		IM_2	T_6	`nM3`	m_6	T_8
	FM2		T_7	`M1`	m_7	T_8
	MM		T_8	`M2`	m_8	T_7, T_9
	MD		T_9	`Mf`	*nil*	T_9

Table 4.3: Algorithm `FormHexMatter` designed to solve the HexMF problem. To task T_i has associated a predicate P_i as shown in Equation (3.1). To recognize the task to perform, each robot evaluates the predicates starting from P_9 and proceeding in the reverse order until a true precondition is found.

T_1 the two most internal robots move toward each other according to m_1, hence the same considerations as above hold until their distance reduces to D and a molecule is formed, see Figure 4.4.right.

The reached configuration in Figure 4.4.right belongs to T_3. In fact, `Mf` is clearly false, that is the configuration is not in T_9. As `FarC` is false, then `M1`, `nM3`, `nM2`, and `nM1` are false, that is the configuration in not in $T_7, \ldots, T_4$, respectively. For the same reason, the embedding of G_H cannot be defined and hence $|Mat| = 0$ and the configuration is not in T_8. Concerning $\texttt{iM3}''$, circle C_R contains only two robots, hence the predicate is false. Concerning $\texttt{iM3}'$, the distance from $c(R)$ does not identify exactly three robots, hence the predicate is false too. Since $|Mol| = 1$ and not all the robots are on C^* then `cM` is true, that is the configuration belongs to T_3. In Figure 4.4 also the trajectories traced by the robots are shown during T_3 and the above Boolean values hold until the last robot reaches C^*. In particular, for $\texttt{iM3}'$ it is possible that at some point the distance from $c(R)$ identifies exactly three robots, however, in that case $\rho(R') < 3$.

The reached configuration in Figure 4.7.left belongs to T_4. In fact, here $Mat = \emptyset$, hence `Mf`, `M2`, `M1`, are false and the configuration is not in T_9, T_8, nor T_7. As $|Mol| = 1$ then `nM3` and `nM2` are false, that is the configuration does not belong to T_6 nor T_5. Since $|Mat| = 0$, `FarC` is true, $|Mol'| = 1$ and there are no molecules on C^* then `nM1` is true and the configuration is in T_4. The corresponding move m_4 makes robots on p_1 and p_2 rotate clockwise on C^* until forming two molecules. During the movements, it is possible that one molecule appears before the other but this does not affect the truth value of the above predicates.

The reached configuration in Figure 4.7.right belongs to T_5. In fact, here $Mat = \emptyset$, hence `Mf`, `M2`, `M1`, are false and the configuration is not in T_9, T_8, nor T_7. As $|Mol| = 3$ then `nM3` is false, that is the configuration does not belong to T_6. Since $|Mat| = 0$, `FarC` is true, $|Mol| = 3$ with 1 or 2 molecules on C^* and 1 internal, and the number of molecules on C^* is no less than $\rho(R)$ then `nM2` is true and the configuration is in T_5. Here the internal molecule radially moves along L reaching the side of a hexagon of side D centered in $c(C^*)$, hence making $|Mat| = 1$. During the movement, the truth value of the above predicates is not affected.

The reached configuration in Figure 4.10 belongs to T_8. In fact, here `Mf` is false, that is the configuration is not in T_9. Since $|Mat| > 0$, $|Mol \setminus Mat| = 2$, and $\rho(Mat) = 1$, then `M2` holds and the configuration is in T_8. Move m_8 involves the two external molecules, one by one, and leads them to be part of the matter. During the movement and after the first molecule arrives, $|Mol \setminus Mat| = 1$ and $\rho(R \setminus \{r_1, r_2, r_3\}) = 1$, hence the truth value of the above predicates is not affected.

The reached configuration in Figure 4.9.left belongs to T_7. In fact, here `Mf` is false, that is the configuration is not in T_9. $|Mol| \setminus |Mat| = 0$, that is `M2` is false the configuration is not in T_8. Since $|Mat| > 0$, $|Mol \setminus Mat| < 3$ and `FarC` is true then `M1` holds and the configuration is in T_7. By alternating tasks T_7 and T_8 the final configuration in Figure 4.3.left is achieved. According to precondition `Mf`, the final configuration belongs to Task T_9, where only the *nil* movement is performed. According to Theorem 5, the reached configuration admits symmetricity 1 whereas the initial configuration of Figure 4.4.left has symmetricity 2. In fact, this is possible since $\rho(\mu) = 2$.

By considering the configurations shown in Figures 4.5 and 4.6 (both referring to configurations of symmetricity three) it is also possible to simulate the evaluation of all the predicates to see that in such configurations $P_9, P_8, \ldots, P_3$ are all false while P_2 holds (thanks to `iM3′` and `iM3″`, respectively). During the execution of the algorithm, these configurations will be processed by tasks T_3 and T_4 until reaching the configuration shown in Figure 4.8. It can be observed that, in such a configuration, predicates P_9, P_8, and P_7 are all false while P_6 holds.

According to the definition of all the Boolean variables given in Section 4.3.2 and to the above examples, we can make the following remark.

Remark 11. *Algorithm* ***`FormHexMatter`*** *fulfills Property* Prop_1.

Using the algorithm in the Compute phase. According to Lemma 14 and Remark 11, we get that all properties Prop_1, Prop_2, and Prop_3 hold. As a consequence, `FormHexMatter` can be used in the `Compute` phase as follows:

> *– if any robot r (or molecule μ) executing algorithm* ***`FormHexMatter`*** *detects that predicate P_i holds, then r (or μ) simply performs the move m_i associated with task T_i.*

Correctness

In this section, we formally prove that algorithm `FormHexMatter` solves the HexMF problem. According to the methodology proposed in [44] and reported in Section 3.2, the correctness of the proposed algorithm can be obtained by proving that all the following properties hold:

H_1: *The algorithm never generates unsolvable configurations. According to Theorem 5, this implies that each configuration $R(t)$, $t > 0$, generated by the algorithm fulfills $\rho(R(t)) \leq 3$.*

H_2: *The movement of each robot is collision-free (cf. Remark 9).*

H_3: *For each task T_i, the transitions from T_i to any other task are "exactly" those declared in Table 4.3; moreover, all such transitions lead to stationary configurations.*

H_4: *Each transition in Table 4.3 occurs after a finite number of cycles. This means that the generated configurations can remain in the same task only for a finite number of cycles.*

Since these properties must be proved for each transition/move, then in the following we provide a specific lemma for each task. Property H_3 does not directly implies that robots/molecules "complete" each task in a finite amount of time. In fact, there is a cycle created by transitions between tasks T_7 and T_8. Anyway, a final theorem will assess the correctness of `FormHexMatter` by making use of all the proved properties H_1–H_4 for each task and by also showing that there is a finite number of transitions between tasks T_7 and T_8.

Lemma 15. *Let R be a stationary configuration in T_1. From R, `FormHexMatter` eventually leads to a stationary configuration belonging to T_3 or T_4.*

Proof. Let us analyze properties H_i, for $1 \leq i \leq 4$, separately.

H_1: In this task, only configurations with symmetricity of 1 or 2 are processed. In the particular case in which there is a robot in $c(R)$, that robot is moved away to create an asymmetric configuration. As a consequence, there are always two robots r_1 and r_2 detectable, and such robots are those closest to $c(R)$. Robots r_1 and r_2 are moved toward the circle C_m so that exactly one molecule is created, eventually. Hence, the symmetricity always remains at most 2 until the end of the task.

H_2: There are at most two robots r_1 and r_2 moving toward C_m. When their reciprocal distance becomes D, if a molecule is formed no collision can occur. When they are at distance D, a molecule is not formed only if either constraint

$\mathcal{C}_2$ or constraint $\mathcal{C}_3$ are not satisfied (cf. Section 4.1). In these cases the two robots could potentially reach the same point on C_m and collide. However, condition $\mathcal{C}_2$ cannot occur since there are no further robots close to $c(R)$ and the starting distance between robots is more than $2D$. Regarding condition $\mathcal{C}_3$, there should be a third robot r_3 at distance D from either r_1 or r_2 when $d(r_1, r_2) = D$. In this case, it is not difficult to prove that r_3 is closer to $c(R)$ than r_2, and this is a contradiction for the definition of r_1 and r_2.

H_3: We show that each configuration generated from R remains in T_1 until the moving robots r_1 and r_2 have reached their targets. Since R belongs to T_1, then the precondition of T_i, for each $i > 1$, is false with respect to R. In particular, since the precondition of T_8 is false, then in the considered configuration R there are no molecules. As a consequence, since all preconditions of $T_9, T_8, \ldots, T_3$ require formed molecules to hold, they remain false until at least a molecule if formed. Concerning T_2, since its precondition is false in R, then it remains as such during the movements of r_1 and r_2. In particular, $\texttt{iM3}''$ is false because when the two robots start moving there remain less than 3 robots on C_1^R, and $\texttt{iM3}''$ is incompatible with the kind of movement performed by r_1 and r_2.

We now show that when r_1 and r_2 reach their targets, a stationary configuration R' belonging to either T_3 or T_4 is generated. In fact, when r_1 and r_2 reach their targets, preconditions of $T_9, T_8, \ldots, T_5$ do not hold because they require $|Mat| > 0$ or $|Mol| > 1$, against the presence of just one molecule. The membership R' depends on `FarC` only: if `FarC` is false, then the obtained configuration is stationary in T_3 because there is only one molecule and both $\texttt{iM3}'$ and $\texttt{iM3}''$ are false for the same reasons above; when `FarC` holds, since there are no molecules on C^*, then R' is a stationary configuration in T_4.

H_4: As long as the configuration remains in T_1, the distance of each moving robot from C_m decreases. Hence, within a finite number of computational cycles, the robots create a molecule.

□

Lemma 16. *Let R be a stationary configuration in T_2. From R, `FormHexMatter` eventually leads to a stationary configuration belonging to T_3 or T_4.*

Proof. Let us analyze properties H_i, for $1 \leq i \leq 4$, separately.

H_1: In this task, there are always three robots detectable, in such a way that without them the configuration admits a symmetricity of 3. The defined move guarantees that r_1, r_2, r_3 are the closest robots to the targets that accomplish the task. Moreover, while such robots move, they decrease their distances from their target and hence they are always distinguishable. It follows that symmetricity never overcomes 3.

H_2: When iM3′ holds, r_1, r_2, and r_3 rotate clockwise until either they create three molecules, or they become aligned with robots r'_1, r'_2, and r'_3, respectively, located on C_2^R. In this second case, they radially move toward r'_1, r'_2, and r'_3, again creating three molecules. Hence, no collisions are possible. When iM3″ holds, r_1, r_2, r_3 rotate clockwise until creating three molecules with three robots located on C_1^R or C_2^R. Again, since the starting distance between robots is more than $2D$, by such a movement no collisions are possible. Furthermore, the moving robots cannot meet other molecules along their trajectory as the configuration did not contain any molecule when the task started.

H_3: We show that each configuration generated from R remains in T_2 until all the three moving robots r_1, r_2 and r_3 have reached their targets. Since R belongs to T_2, then the precondition of T_i, for each $i > 2$, is false with respect to R. In particular, since the precondition of T_8 is false, then there are no molecules in R. As a consequence, since all preconditions of $T_9, T_8, \ldots, T_3$ require the presence of molecules to hold, they remain all false until at least one molecule if formed. Since iM3′ and iM3″ remain valid until r_1, r_2, and r_3 are all part of a molecule, then each obtained configuration remains in T_2 until one molecule is formed.

Let R' be any configuration observed in the interval of time in which one or two molecules are formed. During such an interval, the following properties hold in R': (1) variable FarC is false (this can be easily observed since the center of the minimum circle enclosing all the formed molecules does not coincide with $c(R)$, as requested by the definition of C^*), (2) $|Mat| = 0$ (the formed molecules are not positioned as the definition of matter requires), and (3) both iM3′ and iM3″ remain valid. As a consequence of these properties, R' is evaluated as belonging to T_2.

Let R'' be any configuration observed as soon as the three moving robots reach their targets. R'' does not belong to T_9, T_8 and T_7 since otherwise $|Mat| > 0$ (and this is false since the three formed molecules are not relatively positioned as the definition of matter requires). It does not belong to T_6, since it requires $|Mol| = 6$ against the only three formed molecules. R'' is not in T_5, since it requires molecules formed on C^* against the only three molecules formed close to $c(R)$. Finally, the membership of R'' depends on FarC only: if FarC is false, then R'' is a stationary configuration in T_3 because there are only three molecules and both iM3′ and iM3″ are false since r_1, r_2, and r_3 all belong to molecules; if FarC holds, since there are no molecules on C^*, then R'' is a stationary configuration in T_4.

H_4: As long as the configuration remains in T_3, the distance of each moving robot from its target decreases. Hence, within a finite number of computational cycles, each moving robot reaches its target.

□

Lemma 17. *Let R be a stationary configuration in T_3. From R,* `FormHexMatter` *eventually leads to a stationary configuration belonging to T_4.*

Proof. During task T_3, we observe the following properties concerning move m_3. In the first part, the robots move radially outward, so the mutual distances of the robots inside C^* can only increase. In the second part of the movement, the robots rotate clockwise on a circle in the center of the ring but remaining within different sectors; being at a distance greater than D from C^* and in different sectors, these robots cannot form molecules either during rotation or during movement toward the target.

We can now analyze properties H_i, for $1 \leq i \leq 4$, separately.

H_1: As there are one or three molecules defining $c(C^*)$ and as no further molecules are created according to the above observation, then $\rho(R)$ can be at most 3.

H_2: Each moving robot r is tracing a trajectory suitably defined to not incur in collisions nor create molecules as described in the above observation.

H_3: We show that each configuration generated from R remains in T_3 until all the moving robots have reached the targets. In R, $\texttt{pre}_3 = \texttt{cM}$ holds whereas $\texttt{pre}_i$, $i > 3$, does not hold. This implies that `FarC` is false (derived from `cM` true) and $|Mat| = 0$ (derived from $\texttt{pre}_8 = \texttt{M2}$ false). Since the value of these two variables is not affected by the robots' movement, then each configuration R' obtained before all robots reach the targets does not belong to $T_9, T_8, \ldots, T_4$. Moreover, since it can be easily observed that `cM` still holds in R', then R' remains in T_3.

Let R'' be the configuration obtained at the time in which all the moving robots reach the targets. Since the molecules in R did not change their position, then still $|Mat| = 0$ in R''; moreover, non new molecules have been created in R''. This implies that R'' is not in $T_9, T_8, \ldots, T_5$. Finally, since `FarC` became true in R'', then R'' results to be a stationary configuration in T_4.

H_4: As long as the configuration remains in T_3, the distance of each moving robot from its target decreases. Hence, within a finite number of computational cycles, the robot reaches its target.

□

Lemma 18. *Let R be a stationary configuration in T_4. From R,* `FormHexMatter` *eventually leads to a stationary configuration belonging to T_5 or T_6.*

Proof. Let us analyze properties H_i, for $1 \leq i \leq 4$, separately.

H_1: During the robots' movement there are only one or three molecules inside C^* defining $c(C^*)$, hence $\rho(R)$ can be at most 3.

H_2: Each moving robot r is going along C^* toward the next closest robot r' in the clockwise direction. Hence, conditions $\mathcal{C}_2$ and $\mathcal{C}_3$ cannot occur since no robots are in between r and r' and once $d(r, r') = D$ a molecule is created.

H_3: Let R' be any configuration observed during the robots' movement and before all moving robots reach their targets. In both R and R' we have $|Mat| = 0$ (as remarked in the proofs of the previous lemmas, the molecules inside C^* are not positioned as the definition of matter requires). This implies that `Mf`, `M2` and `M1` remain false and hence R' is not in T_9, T_8, T_7. During the movement, predicates `nM2` and `nM3` are false due to the non-consistent numbers of molecules inside and on C^*. Hence, R' remains in T_4.

Let R'' be the configuration observed as soon as all the moving robots reach their targets. In R'' there can be three, two or one molecule on C^*. If three, it means there are three molecules inside C^* and predicate `nM3` becomes true, i.e., the configuration is in T_6 and it is stationary. If one or two, there is only one molecule inside C^* and predicate `nM2` becomes true, whereas `nM3` is false. Hence, the configuration is in T_5 and it is stationary.

H_4: As long as the configuration remains in T_4, the distance of each moving robot from its target decreases. Hence, within a finite number of computational cycles, each moving robot reaches its target.

□

Lemma 19. *Let R be a stationary configuration in T_5. From R,* `FormHexMatter` *eventually leads to a stationary configuration belonging to T_8.*

Proof. Let us analyze properties H_i, for $1 \leq i \leq 4$, separately.

H_1: As soon as the molecule inside C^* moves, $\rho(R)$ can only be equal to 1.

H_2: Since there is only one molecule inside C^*, no collisions among robots are possible, when the molecule is moving. Similarly, no other molecules are met.

H_3: Let μ be the moving molecule, and let R' be any configuration observed during the movement of μ and before μ reaches its target. Of course, $|Mat| = 0$ in R'. Hence, both `M1` and `M2` are false in R' and hence R' is not in T_9, T_8, T_7. R' is not in T_6 because `nM3` is false (there is only μ inside C^*). Since `nM2` remains true, R' is in T_5.

Let R'' be the configuration observed as soon as μ reaches the target. In this configuration we have $|Mat| = 1$. As a consequence, R'' is not in T_9. `M2` holds in R'' ($|Mat| = 1$, the matter formed does not admit a rotation, and $|Mol \setminus Mat| > 0$). Since there is at least one molecule on C^*, then R'' results to be a stationary configuration in T_8.

H_4: As long as the configuration remains in T_5, the distance of the moving molecule from its target decreases. Hence, within a finite number of computational cycles, the molecule reaches its target.

□

Lemma 20. *Let R be a stationary configuration in T_6. From R,* `FormHexMatter` *eventually leads to a stationary configuration belonging to T_8.*

Proof. Let us analyze properties H_i, for $1 \leq i \leq 4$, separately.

H_1: Since there are three molecules on C^*, $\rho(R)$ can only be 1 or 3.

H_2: Since there are only molecules moving, collisions among robots cannot occur. Moreover, by move m_6, molecules always move on radial trajectories toward $c(C^*)$ without ever touching each other.

H_3: Let R' be any configuration observed during the molecules' movement and before they all reach their targets. R' is clearly not in T_9 since there are still molecules to be added to the matter. Moreover, Property 2 in the definition of `M2` is false and hence R' does not belong to T_8. Since there are three molecules on C^*, then $|Mol \setminus Mat| > 3$ and hence the last property of `M2` does not hold: hence, R' is not in T_7. Since `nM3` is clearly not affected by the movements, then R' remains in T_6.

Let R'' be the configuration observed as soon as each molecule reaches its target. Again, R'' is not in T_9 since there are still molecules to be added to the matter. `M2` is true in R'' (in fact, $|Mol \setminus Mat| = 3$), and hence R'' results to be a stationary configuration in T_8.

H_4: As long as the configuration remains in T_6, the distance of each moving molecule from its target decreases. Hence, within a finite number of computational cycles, each molecule reaches its target.

□

Lemma 21. *Let R be a configuration with $|Mat| > 0$. Then, $\rho(R)$ can be either 1 or 3.*

Proof. By construction, the molecules constituting the first level of the matter are three at most. If they are less than three then $\rho(R) = 1$. Else, if they are three, then $\rho(R)$ cannot be larger than three nor equal to two. □

Lemma 22. *Let R be a stationary configuration in T_7. From R,* `FormHexMatter` *eventually leads to a stationary configuration belonging to T_8.*

Proof. Let us analyze properties H_i, for $1 \leq i \leq 4$, separately.

H_1: Follows directly from Lemma 21.

H_2: Each moving robot r is going along C^* toward the next closest robot r' in the clockwise direction. Hence, conditions $\mathcal{C}_2$ and $\mathcal{C}_3$ cannot occur since no robots are in between r and r' and once $d(r, r') = D$ a molecule is created.

H_3: Let R' be any configuration observed during the robots' movement and before they all reach their targets. R' is clearly not in T_9 since there are still molecules to be added to the matter. If only one robot is allowed to move, then `M2` remains false until one molecule is created. If three robots are allowed to move, it means that the matter admits a rotation and $\rho(R' \setminus \{r_1, r_2, r_3\}) = 3$ while $|Mol \setminus Mat| < 3$, hence again `M2` is false. Summarizing, in both cases (one or three robots moving) we get that R' belongs to T_7.

Let R'' be the configuration observed as soon as each robot reaches its target. Again, R'' is not in T_9 since there are still molecules to be added to the matter. `M2` is true in R''. In fact, either $|Mol \setminus Mat| = 3$ or $|Mol \setminus Mat| = 1$; in the latter case either the matter does not admit a rotation or $\rho(R \setminus \{r_1, r_2, r_3\}) = 1$. Hence, R'' is a stationary configuration in T_8.

H_4: As long as the configuration remains in T_7, the distance of each moving robot from its target decreases. Hence, within a finite number of computational cycles, the robot reaches its target.

□

Lemma 23. *Let R be a stationary configuration in T_8. From R, `FormHexMatter` eventually leads to a stationary configuration belonging to T_7 or T_9.*

Proof. Move m_8 aims to bring molecules formed on C^* to join the matter. Hence, during this task, predicates `FarC` and $|Mat| > 0$ remain true. Let us analyze properties H_i, for $1 \leq i \leq 4$, separately.

H_1: Follows directly from Lemma 21.

H_2: Since there are only molecules moving, collisions among robots cannot occur. Moreover, by move m_8, molecules always move on free trajectories.

H_3: Let R' be any configuration observed during the molecules' movement and before they all reach their targets. In R', `M2` remains trivially true as long as $|Mol \setminus Mat| = 3$. Otherwise `M2` remains true since, if $|Mol \setminus Mat| = 1$, then there is exactly one molecule inside C^* and hence $\rho(R \setminus \{r_1, r_2, r_3\}) = 1$; if $|Mol \setminus Mat| = 2$, then $|Mat| \mod 3 = 2$, and hence the matter does not admit a rotation. It follows that R' remains in T_8.

Let R'' be the configuration observed as soon as each molecule reaches its target. In R'', it can be easily observed `M2` becomes false (because $|Mol \setminus$

$Mat| = 0$) and `M1` becomes true. Then, if no further robots are on C^*, then R'' is a stationary configuration in T_9. Otherwise, if there are still robots on C^*, then R'' is a stationary configuration in T_7.

H_4**:** As long as the configuration remains in T_8, the distance of each moving molecule from its target decreases. Hence, within a finite number of computational cycles, the molecule reaches its target.

□

Theorem 6. *Let $m > 3$ be an integer and let R be any initial configuration composed of $|R| = 2m$ asynchronous robots moving on the plane. If robots in R have a pairwise distance greater than $2D$, then* `FormHexMatter` *correctly solves the HexMF problem.*

Proof. Lemmata 15-23 ensure that properties H_1, H_2, H_3, and H_4 hold for each task T_1, $T_2, \ldots, T_9$. Then $\rho(R(t))$, for $t > 0$, is always less than or equal to 3; the moves of the robots are all collision-free; all the transitions are those reported in Table 4.3; and the generated configurations can remain in the same task only for a finite number of cycles. Lemmata 15-23 also show that from a given task only subsequent tasks can be reached, or `Mf` eventually holds (and hence HexMF is solved). The only exception is the cycle among tasks T_7 and T_8. However, in this case, at the end of T_8, the number of molecules composing the matter increases, and since no molecule is moved away from the matter, task T_9 is reached from T_8 after a finite number of transitions between T_7 and T_8. This formally implies that, for each initial configuration R and for each execution $\mathbb{E} : R = R(0), R(1), R(2), \ldots$ of `FormHexMatter`, there exists a finite time $t' > 0$ such that $R(t')$ is similar to the matter to be formed in the HexMF problem and $R(t) = R(t')$ for each time $t \geq t'$. □

4.3.4 The Moblot model extends Oblot

Exploiting the definition of the HexMF problem provided in the case study, we can formally prove that $\mathcal{MOBLOT}$ is an extension of $\mathcal{OBLOT}$. To this aim, we first introduce some notation. Given two robot models $\mathcal{M}_1$ and $\mathcal{M}_2$, inequality $\mathcal{M}_1 \geq \mathcal{M}_2$ means that any problem that can be solved in $\mathcal{M}_2$ is also solvable in $\mathcal{M}_1$ (i.e., $\mathcal{M}_1$ is not less powerful than $\mathcal{M}_2$). Inequality $\mathcal{M}_1 > \mathcal{M}_2$ means that $\mathcal{M}_1 \geq \mathcal{M}_2$ holds and there exists a problem that can be solved in $\mathcal{M}_1$ but not in $\mathcal{M}_2$ (i.e., $\mathcal{M}_1$ is more powerful than $\mathcal{M}_2$). We can prove the following result.

Theorem 7. $\mathcal{MOBLOT} > \mathcal{OBLOT}$.

Proof. We first show that $\mathcal{MOBLOT} \geq \mathcal{OBLOT}$. To this aim, observe that in case each molecule in $\mathcal{M}$ is constituted by a single robot without any extent nor extra capabilities, then $\mathcal{MOBLOT}$ reduces to $\mathcal{OBLOT}$.

We now show that $\mathcal{MOBLOT} > \mathcal{OBLOT}$. Consider HexMF, and let $\mathcal{I}$ be the set of all possible instances $(R, \mathcal{M}, \mathcal{F})$ for HexMF fulfilling the conditions in Theorem 5.

Let us analyze whether there exists an algorithm $\mathcal{A}'$ defined according to the $\mathcal{OBLOT}$ model and able to solve HexMF for each instance in $\mathcal{I}$. In other words, we are asking whether $\mathcal{A}'$ is able to form some pattern in $\mathcal{F}$ by moving robots from R *always assuming them as single units, that is without exploiting the capabilities of molecules in* $\mathcal{M}$. It is well known from [131] that "a pattern F cannot be formed from a configuration R when $\rho(R)$ does not divide $\rho(F)$". Since there are configurations $(R, \mathcal{M}, \mathcal{F}) \in \mathcal{I}$ such that $\rho(R) = 2$ and $\rho(F) \in \{1, 3\}$ for each $F \in \mathcal{F}$, then $\mathcal{A}'$ cannot solve the problem. Hence, HexMF cannot be solved in the $\mathcal{OBLOT}$ model.

Consider now `FormHexMatter`, the algorithm described in Section 4.3.2 and formalized in Section 4.3.3. Theorem 6 shows that `FormHexMatter` is able to solve HexMF for each instance in $\mathcal{I}$ by suitably exploiting the molecules' capabilities. In fact, the molecules have the ability to break the symmetry of the original configuration, provided that at least one molecule shows this symmetry. In particular, when the first property of Theorem 5 does not hold but the second does, algorithm `FormHexMatter` creates a single molecule close to the center of the configuration and after moves it to break the initial symmetry. □

4.4 The Moblot model on grid graphs

In this section, we apply the $\mathcal{MOBLOT}$ model to **synchronous** robots moving on a square grid and in what follows, we revise the properties characterizing the $\mathcal{MOBLOT}$ model introduced for robots moving on the Euclidean plane. On grids, robots and molecules can move along the grid lines and only toward an adjacent grid point in each step. The investigation on grids-based terrains is motivated by the fact that they are used in real life robotic navigation systems. From an algorithmic point of view, the restrictions imposed by the environment on movements make more difficult to solve problems. We also present an extension of Theorem 5 to take into account the underlying grid-based environment. A $\mathcal{MOBLOT}$ system is composed by a set $R = \{r_1, r_2, \ldots, r_n\}$ of n robots, that live and operate in graphs. As chirality is assumed, and we are considering a regular square grid as a field of movement, the only possible symmetries are rotations of 90 or 180 degrees.

A molecule μ is specified by a **fixed pattern** defined with respect to the regular square grid. For instance, in what follows, we consider robots to form a molecule if they are disposed as a possible polyomino that is a geometric shape composed connecting squares of the same size orthogonally (i.e., at the edges and not the corners, cf., Fig. 4.11) [81]. Each square has the same dimensions of the squares of the grid and it is centered on one of the robots composing the molecule. Once a molecule is formed, it is assumed to have an extent $B(\mu)$ given by the union of the squares of which a polyomino is composed. Fig. 4.11.left shows a molecule as formed by four

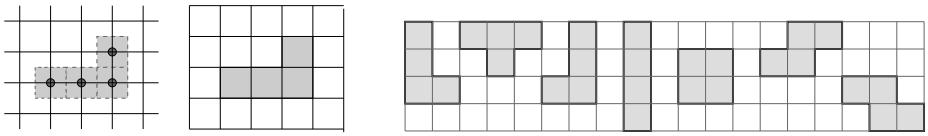


Figure 4.11: Left: A molecule formed by four adjacent robots; middle The dual grid, where only the extent of the molecule is shown; right: all possible tetrominoes. According to their shape, they can be referred to as `L`, `T`, `J`, `I`, `O`, `S`, and `Z`, respectively

robots whereas Fig. 4.11.middle shows the same molecule represented in the **dual grid**, where - for the sake of simplicity - only the extent of the molecule is considered. By $\mathcal{M} = \{\mu_1, \mu_2, \ldots, \mu_m\}$ we denote the set containing all kinds of molecules that can be potentially formed. For example, if the number of squares composing a polyomino equals five, $\mathcal{M}$ is the set of the eighteen possible pentominoes. Regarding the constrains $\mathcal{C}_1, \ldots, \mathcal{C}_5$ defined for the $\mathcal{MOBLOT}$ model in Section 4.1, we specify property $\mathcal{C}_1$ for robots moving on grid graphs.

$\mathcal{C}_1$**:** in initial configurations, each pair of robots is at distance not less than 2, where the distance is the number of edges composing the shortest path connecting them.

The basic properties of such new entities can still be modeled as in $\mathcal{OBLOT}$ systems (and its variants), with the main exception that a molecule not only can move by following an edge of the grid but it also may rotate of 90 degrees with respect to one of the vertices occupied by the robots composing it.

Each type of molecule in $\mathcal{M}$ is provided as input to the algorithm, and the algorithm is responsible to assemble all the molecules so that a more complex structure (i.e., the *matter*) is formed. Also the matter to be formed must be given as input to robots and it can be defined either as according to some adjacency properties or by providing a specific pattern made of molecules (cf., Fig 4.12.right). For ease of the discussion, we refer to the latter case as the **Molecular Pattern Formation** (MPF) problem. In $\mathcal{MOBLOT}$, a robot r performing the `Look` phase is able to detect not only all the other robots but also any formed molecule μ. We denote by *Mol* the set of molecules detected at a given time by a robot r.

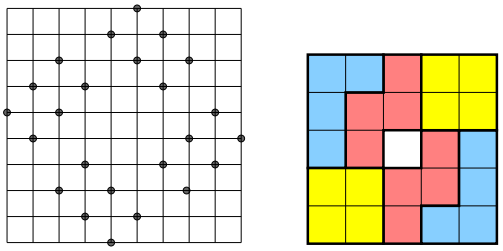


Figure 4.12: Left: a configuration containing only robots with $\rho(C) = 4$. Right: A pattern F with $\rho(F) = 2$ on the dual grid (colors are used for better viewing only).

4.5 The molecular pattern formation problem

In this section, we first give all the necessary concepts and notation needed to formalize the MPF problem and then we state a necessary condition for its feasibility. In the following, we use the term **entity** when we do not need to distinguish between robot and molecule.

Configurations. A square tessellation of the Euclidean plane is the covering of the plane using squares of side length 1, called tiles, with no overlaps and in which the corners of squares are identically arranged. Let S be the infinite lattice formed by the vertices of the square tessellation. The graph G_S is called **grid graph**, its vertices are the points in S and its edges connect vertices that are distance 1 apart.

Let $R = \{r_1, r_2, \ldots, r_n\}$ be the set of robots. The topology where robots are placed is the grid graph $G_S = (V, E)$. A function $\lambda : R \to V$ maps each robot to the vertex in G_S where the robot is placed. Assume that each robot knows the set of available molecules $\mathcal{M}$ and the set $\mathcal{F}$ of possible patterns describing the matter to form. As said above, during the `Look` phase, each robot detects in the local coordinate system both the robots' positions and the set Mol of already formed molecules. We call $C = (G_S, R, \lambda, Mol)$ a **configuration**. Constraint $\mathcal{C}_1$ imposes $Mol = \emptyset$ in each initial configuration.

Symmetricity of a configuration. The concept of graph isomorphism introduced in Section 3.1.1 can be extended to configurations in a natural way. Two configurations $C = (G_S, R, \lambda, Mol)$ and $C' = (G_S{}', R', \lambda', Mol')$ are isomorphic if there exists an isomorphism φ between G_S and $G_S{}'$ that can be extended to obtain a bijection from $V \cup R$ to $V' \cup R'$ such that:

- two robots can be associated by φ only if they reside on equivalent vertices: if $\varphi(r) = r'$ then $\varphi(\lambda(r)) = \lambda'(r')$;
- it preserves molecules: if $\mu = \{r_{i_1}, \ldots, r_{i_t}\} \in Mol$ then $\{\varphi(r_{i_1}), \ldots, \varphi(r_{i_t})\} = \mu' \in Mol'$ and $\mu' = \mu$.

In this way, analogously to graph automorphism, an automorphism of a configuration $C = (G_S, R, \lambda, Mol)$ is an isomorphism from C to itself, and the set of all automorphisms of C forms a group under the composition operation that we call automorphism group of C and denote as $\mathrm{Aut}(C)$. Moreover, if $|\mathrm{Aut}(C)| = 1$ we say that C is **asymmetric**, otherwise it is **symmetric**. Two distinct robots r and r' in a configuration C are **equivalent** if there exists $\varphi \in \mathrm{Aut}(C)$ such that $\varphi(r) = r'$. Note that, the notion of equivalence also applies to molecules.

Remark 12. *Let $C = (G_S, R, \lambda, Mol)$ be a symmetric configuration, $\mathcal{A}$ be any algorithm acting on C, and E be any maximal subset of pairwise equivalent entities in C. Any move planned by $\mathcal{A}$ for an element of E applies to* all *set E.*

As chirality is assumed, it is easy to see that any configuration C defined on G_S admits one type of automorphisms only: **rotations**. A rotation is an isometry defined by a center c and a minimum angle of rotation $\alpha \in \{90, 180, 360\}$ working as follows: if the configuration is rotated around c by an angle α, then a configuration coincident with itself is obtained. The **order** of a configuration is given by $360/\alpha$. A configuration is **rotational** if its order is 2 or 4. The **type of center** of a rotational configuration C is denoted by $tc(C)$ and is equal to:

- 1, when the center of rotation is on a vertex of G_S (see Fig. 4.12.right represented by the dual grid);
- 2, when the center of rotation is on a median point of an edge of G_S;
- 3, when the center of rotation is on the center of a square of the tessellation forming G_S (see Fig. 4.12.left).

The **symmetricity** of a configuration C, denoted as $\rho(C)$, is equal to its order unless its center is occupied by one entity, in which case $\rho(C) = 1$. It comes out that when the configuration is constrained on G_S, then $\rho(C) \in \{1, 2, 4\}$.

We defined $\rho()$ and $tc()$ for any configuration $C = (G_S, R, \lambda, Mol)$ regardless whether Mol is empty or not. Concerning notation, we use $\rho(F)$ and $tc(F)$ to refer to any configuration forming a pattern $F \in \mathcal{F}$. As a special case, we use $\rho(\mu)$ and $tc(\mu)$ to refer to robots within a single molecule μ only. Moreover, for a given pattern $F \in \mathcal{F}$, let $Mol(F)$ denote the set of molecules that form F; clearly, each molecule in $Mol(F)$ also appears in $\mathcal{M}$.

Problem formalization

An **execution** of an algorithm $\mathcal{A}$ from an initial configuration C is a sequence of configurations $\mathbb{E} : C(t_0), C(t_1), \ldots$, where $C(t_0) = C$ and $C(t_{n+1})$ is obtained from $C(t_n)$ by moving some entities according to the result of the `Compute` phase as implemented by $\mathcal{A}$. With respect to the defined $\mathcal{MOBLOT}$ model, the **MPF problem** on the grid graph can be formalized as follows.

Definition 9. *Given an initial configuration $C = (G_S, R, \lambda, Mol = \emptyset)$, a set of molecules $\mathcal{M}$, and a set $\mathcal{F}$ of possible patterns describing the matter to form, the goal is to design a distributed algorithm $\mathcal{A}$ that works for each entity so that eventually they form some pattern in $\mathcal{F}$, if possible. Formally, $\mathcal{A}$ solves the MPF problem for C if, for each possible execution $\mathbb{E} : C = C(t_0), C(t_1), \ldots$ of $\mathcal{A}$, there exists a finite time instant $t_n > 0$ such that in $C(t_n)$ all robots have been assembled into molecules, the molecules form an element in $\mathcal{F}$, and no entity moves after t_n, i.e., $C(t_k) = C(t_n)$ for each $t_k \geq t_n$.*

Theorem 8. *Let $C = (G_S, R, \lambda, Mol)$ be any configuration composed of synchronous robots and $(C, \mathcal{M}, \mathcal{F})$ be an instance of the MPF problem. If there exists an algorithm $\mathcal{A}$ able to form a pattern $F \in \mathcal{F}$ from C, then one of the following holds:*

1. *$\rho(C)$ divides $\rho(F)$ and $(\rho(C) > 1 \Rightarrow tc(C) = tc(F))$;*
2. *$\exists\ \mu \in Mol(F)$: $\rho(C)$ divides $\rho(\mu)$ and $(\rho(C) > 1 \Rightarrow tc(C) = tc(\mu))$.*

Proof. Assume that $\mathcal{A}$ is able to form F without preliminarily forming molecules (i.e., there exists a time $t_n > 0$ such that $C(t_n)$ is similar to F and no molecule is formed in $C(t_i)$ for each $t_i < t_n$). In this case, we have from [131] that property (1) holds. In what follows we assume that $\mathcal{A}$ must create and move some molecules to form F. We also assume $\rho(C) > 1$, otherwise both properties (1) and (2) are trivially verified. Let $\mathbb{E} : C = C(t_0), C(t_1), \ldots$ be the execution of the algorithm $\mathcal{A}$, according to Remark 12, $\rho(C(t_0))$ pairwise equivalent robots move synchronously. Let $C(t_k)$, $k > 0$, be the first configuration containing molecules.

If $C(t_k)$ contains more than one molecule, according to the synchronous moves and to the symmetricity of C, then (i) in $C(t_k)$ there are $\rho(C(t_0))$ molecules, (ii) the molecules in $C(t_k)$ are all equal, (iii) $\rho(C(t_k)) = \rho(C(t_0))$, and (iv) the center of $C(t_0)$ and that of the configuration made by the formed molecules coincide. Then, from $C(t_k)$ on, each move planned by $\mathcal{A}$ maintains at least the same symmetricity $\rho(C(t_0))$ and the same type of center until F is formed. Then, $\rho(C(t_0))$ divides $\rho(F)$ and the center of the formed molecules is maintained. Summarizing, property (1) must hold.

If $C(t_k)$ contains just one molecule μ, then it must be formed around the center of the configuration so that $tc(\mu) = tc(C(t_0))$. Moreover, even in this case $\mathcal{A}$ makes $\rho(C(t_0))$ equivalent robots move synchronously, and hence $\rho(\mu)$ must be a multiple of $\rho(C(t_0))$. To summarize, property (2) must hold. □

4.6 The Tetris-like MPF problem

In this section, we introduce **Tetris-Like MPF** (TL-MPF for short), a particular version of the MPF problem. TL-MPF will be used as a case study of the $\mathcal{MOBLOT}$ model to appreciate its facets in grids. In what follows we consider four robots to form a molecule if they are disposed as a possible **tetromino**. The set of all formable molecules, $\mathcal{M}$ is composed by the 7 possible tetrominoes, and each kind of tetromino is denoted by a single character among `L`, `T`, `J`, `I`, `O`, `S`, and `Z`, according to their shape, see Fig. 4.11.right. We recall that a tetromino is formed by 4 robots and that two tetrominoes cannot overlap. We say that two tetrominoes are **adjacent** when robots belonging to distinct tetrominoes are adjacent in G_S. In TL-MPF, $\mathcal{M}$ contains all the seven tetrominoes, $\mathcal{F} = \{F\}$ (where F is any set of four or more tetrominoes), and accordingly to the definition of F, the set of **initial** configurations consists of configurations in G_S having a multiple of 4 and with at least 16 robots. According to constraint $\mathcal{C}_1$, robots are pairwise non-adjacent.

Note that, each initial configuration C with $\rho(C) = 1$ is necessarily asymmetric. This implies that, in each initial configuration C, the symmetricity induces a partition of all the entities in subsets having the following relevant properties: (1) each set has size equal to $\rho(C)$, and (2) in each set, the entities are pairwise equivalent. Each set in this partition is called an **orbit**.

In principle, given an initial configuration C and a pattern F to be formed, it is possible that no algorithm exists for solving TL-MPF. We now specialize Theorem 8 to provide the definition of **potentially-formable patterns** from C. According to $\rho(C)$, we have the following cases:

Corollary 9. *Given a configuration C and a pattern $F \in \mathcal{F}$, F is* potentially-formable *from C if one of the following conditions hold:*

1. $\rho(C) = 1$;
2. $\rho(C) = 2$ *and*
   (a) $tc(C) = tc(F)$ *and* $\rho(F) \in \{2, 4\}$, *or*
   (b) $tc(C) = 2$ *and* $\{\mathtt{S}, \mathtt{Z}, \mathtt{I}\} \cap Mol(F) \neq \emptyset$, *or*
   (c) $tc(C) = 3$ *and* $\mathtt{O} \in Mol(F)$;
3. $\rho(C) = 4$ *and*
   (a) $tc(C) = tc(F)$ *and* $\rho(F) = 4$, *or*
   (b) $tc(C) = 3$ *and* $\mathtt{O} \in Mol(F)$.

In the following, we describe algorithm $\mathcal{A}_{\mathsf{TL}}$ that solves TL-MPF for each pair (C, F), where C is an initial configuration and F is potentially-formable from C. This provides a complete characterization of the feasibility of TL-MPF.

problem	*sub-problems*	*task*	*transitions*
TL-MPF	MS: Make working Space	T_1	T_2, T_3
	Forming $\rho(C)$ new Molecules	T_2	T_4
	SB: forming one central molecule	T_3	T_4
	AM: Adding Molecules to pattern	T_4	T_2, T_5
	Term: problem Termination	T_5	-

Table 4.4: The decomposition of TL-MPF into tasks, with SB standing for subproblem Symmetry Breaking by means of the formation of one central molecule.

The algorithm $\mathcal{A}_{\mathsf{TL}}$ has been designed according to the methodology proposed in [44] and reported in Section 3.2. Table 4.4 (explained later) shows the decomposition into tasks for TL-MPF. $\mathcal{A}_{\mathsf{TL}}$ is responsible for allowing entities to detect which task must be accomplished in any configuration observed during an execution.

In the next section, we give a description of each task T_i, by including details about the corresponding move m_i and precondition $\mathtt{pre}_i$. From the definition of the preconditions, it follows that also Prop_1 holds.

4.6.1 The resolution algorithm

As shown in Table 4.4, the problem has been decomposed into five tasks. To describe $\mathcal{A}_{\mathsf{TL}}$ in detail, some further definitions are required. Let $mbr(R)$ denote the **minimum bounding rectangle** of R, that is the smallest rectangle (with sides parallel to the edges of G_S) enclosing all robots. Note that $mbr(R)$ is unique. By $c(R)$ we denote the center of $mbr(R)$. Similarly, $mbr(F)$ is defined for the minimum bounding rectangle enclosing the molecules forming F. In the following, we use the term *partial-molecule* to refer to a pair of adjacent robots that can be later used to assemble a molecule. Note that, by Property $\mathcal{C}_1$, a partial-molecule cannot exist in initial configurations. We define *point-joined-robots*, the configuration in which two robots are aligned along the diagonal of a cell of the grid and their corresponding monominoes in the dual graph intersect in one point. *Mol* denotes the set of all the molecules formed so far; $F' \subseteq Mol$ is the set of formed molecules that are already assembled to form the matter, i.e., a sub-pattern of F.

The view of the robots. robots encode the perceived configuration into a binary string denoted as $LSS(R)$ (Lexicographically Smallest String) and computed as follows (cf. [4]). They assign a string to each corner of $mbr(R)$: the grid enclosed by $mbr(R)$ is analyzed row by row or column by column - the direction is given by the smallest side of $mbr(R)$ - and 1 or 0 correspond to the presence or the absence, respectively, of a robot for each encountered vertex. From the 4 corners they get up

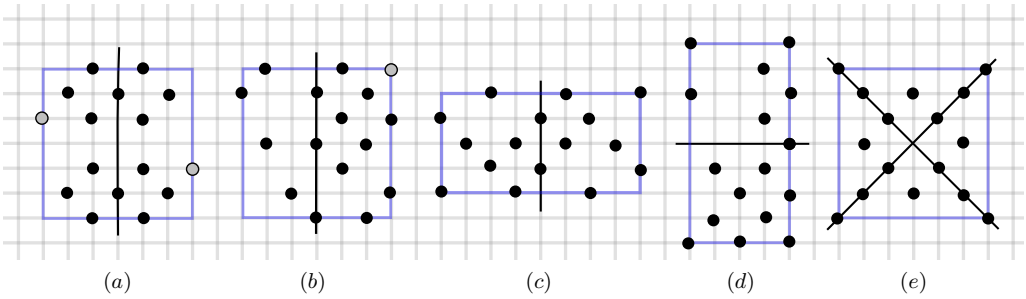


Figure 4.13: Division of $mbr(R)$ into regions based on shape and $\rho(C)$. The $mbr(R)$ is shown in blue. Robots with the minimum view are shown in grey only when needed. In (a): $\rho(C) = 2$, $LSS(R)$ =0000100 0100010 1010101 0100010 1010101 0100010 0010000; in (b): $\rho(C) = 1$, $LSS(R)$=0000010 0001001 0100000 1001010 0010101 1001010 0100101. The $mbr(R)$ is a square and it is partitioned into 2 equal regions by a line passing through $c(R)$ and parallel to the sides of $mbr(R)$ where the robots with minimum view reside; In (c): $\rho(C) = 2$; in (d): $\rho(C) = 1$, $mbr(R)$ is a rectangle and it is partitioned into 2 equal regions by a line passing through $c(R)$ and parallel to the shorter sides; in (e): $\rho(C) = 4$, then $mbr(R)$ is a square and hence it is partitioned by using two diagonals.

to 8 different strings, and the lexicographically smaller one is $LSS(R)$. Note that if two strings obtained from opposite corners along opposite directions are equal, then the configuration is rotational, otherwise it is asymmetric. The robot(s) with **minimum view** is the one with minimum position in $LSS(R)$. The same approach can be used for F but with strings formed by letters (i.e., if the analyzed vertex is occupied by a robot forming molecule `Z`, then `Z` is inserted in the string, otherwise, if the vertex is unoccupied, `X` is inserted).

Tasks T_1 - Make Working Space. The goal of this task is to increase the distance between robots. In fact, in an initial configuration, robots might be too close to each other (e.g., when robots occupy alternatively the vertices of the grid) and the movements might cause the formation of undesired partial-molecules. During T_1, according to Remark 12, robots move away from $c(R)$. At the end of the task, consecutive orbits of robots are at distance at least $\delta = 2$ from each other and there is also an empty space Q in the center of the configuration that contains at most the orbit of robots closest to $c(R)$, see Figure 4.16 for an example. The space Q is a square centered in $c(R)$ and its side is $side(Q) = 2S$, where $S = \max\{w(F), h(F)\}$ and $w(F)$ and $h(F)$ are the width and the height, respectively, of $mbr(F)$. The fixed distance δ guarantees that robots have enough space for moving and creating a molecule.

We now provide all details necessary to formalize the move m_1. The first necessary step is that of dividing $mbr(R)$ into **regions** according to $\rho(C)$, cf. Fig. 4.13. If

$\rho(C) = 4$ then $mbr(R)$ is a square and hence it is partitioned by using two diagonals. If $\rho(C) = 2$ and $mbr(R)$ is a square, it is partitioned into 2 equal regions by a line passing through $c(R)$ and parallel to the sides of $mbr(R)$ where the robots with minimal view reside.[2] If $\rho(C) = 1$ and $mbr(R)$ is a square, it is partitioned into 2 equal regions by a line passing through $c(R)$ and parallel to the side of $mbr(R)$ where the robot with minimum view resides. If $\rho(C) = 1, 2$ and $mbr(R)$ is a rectangle, it is partitioned into 2 equal regions by a line passing through $c(R)$ and parallel to the shorter sides.

Each robot belongs to one of the formed regions, unless it is on a half-line of the lines used for partitioning $mbr(R)$; in this case, the robot belongs to the region to the right of the half-line. Each side ℓ of $mbr(R)$ entirely contained in a region is said to be "associated with" that region.

When $\rho(C) = 2, 4$, robots in each region are numbered as follows. Let ℓ be the side of $mbr(R)$ associated with the region, and v be the leftmost vertex of ℓ. Robots are numbered starting from v, proceeding along ℓ, then continuing in order with all the lines parallel to ℓ. By assuming that the region contains t robots, the first met robot is numbered as r^t and the remaining, in order, as $r^{t-1}, \ldots, r^1$. It is clear that the robots in a region all belong to different orbits and therefore the numbering of robots can be understood as a numbering for the orbits. Hence, orbits are denoted as $O^t, O^{t-1}, \ldots, O^1$.

When $\rho(C) = 1$, the two defined regions may have a different number of robots inside, say t_1 and t_2. Robots are numbered as in the previous cases in both the regions, but they are denoted as $\dot{r}^{t_1}, ..., \dot{r}^1$ in the region containing the robot with minimum view, and as $\ddot{r}^{t_2}, ..., \ddot{r}^1$ in the other region. Hence, orbits are denoted as $\dot{O}^{t_1}, \dot{O}^{t_1-1}, \ldots, \dot{O}^1$, and $\ddot{O}^{t_2}, \ddot{O}^{t_2-1}, \ldots, \ddot{O}^1$. Let $O^t, \ldots, O^1$, with $t = t_1 + t_2$, such that $O^t = \dot{O}^{t_1}, O^{t-1} = \ddot{O}^{t_2}$ and the remaining orbits $O^{t-2}, \ldots, O^1$ are defined by keeping orbits from the two regions in an alternating fashion as long as possible. Let r be a robot in a region associated with a side ℓ, and assume $r \in O^i$: $cd_Q(O^i)$ represents the "current distance" of O^i from Q (that is the distance between r and the side of Q parallel to ℓ), it is negative if r is inside Q; $fd_Q(O^i)$ represents the "final distance" of O^i from Q, that is the distance that robots on O^i must have when the orbit is correctly positioned. These functions are formally defined as follows. When $\rho(C) = 2, 4$:

- $fd_Q(O^1) = \max\{S + 1, cd_Q(O^1)\}$
- $fd_Q(O^i) = \max\{fd_Q(O^{i-1}) + \delta, cd_Q(O^{i-1})\}, \forall i > 1$

When $\rho(C) = 1$:

- $fd_Q(O^1) = \max\{S + 1, cd_Q(O^1)\}$

[2] If the robot with minimum view is on a corner, it is assumed to reside on the clockwise side of $mbr(R)$.

- $fd_Q(O^i) = \max\{fd_Q(O^{i-1}) + \delta, cd_Q(O^{i-1})\}, \forall i > 1, i < t-1$
- $fd_Q(O^t) = fd_Q(O^{t-1}) = \max\{fd_Q(O^{t-2}) + \delta, cd_Q(O^{t-2})\}$

Move m_1 works as follows: *each robot in O^j, for each $j > 1$, moves perpendicularly to the side ℓ of $mbr(R)$ which it is associated to, increasing its distance from $c(R)$, until $cd_Q(O^j) = fd_Q(O^j)$*. Note that the task makes all robots moving concurrently. By defining the two Boolean variables

- $P(k) =$ *each orbit O^i , $i > k$, is correctly positioned with respect to $fd_Q()$*;
- $\mathcal{Q} =$ *square Q is formed with at most one orbit inside*,

it can be observed that task T_1 ends when both $P(1)$ and $\mathcal{Q}$ hold.

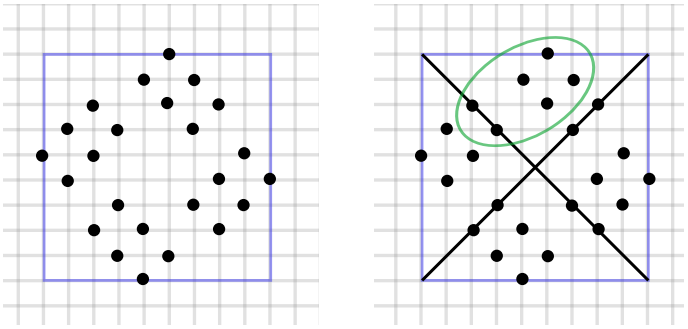


Figure 4.14: Left: an initial configuration C of robots with $\rho(C) = 4$; right: the subdivision into four regions and the subset of six robots belonging to one region.

Task T_2 - Molecules formation. The goal of this task is to create $\rho(C)$ new molecules to add to the matter F' formed so far. Let B' be a Boolean variable that is true when one among the conditions 1, 2.a, and 3.a of Corollary 9 hold. In all these cases, $\rho(F)$ is a multiple of $\rho(C)$, and when $\rho(C) > 1$ then C and F have the same type of center.

To be executed, T_2 requires that B' holds and T_1 is completed (i.e., $P(1)$ and $\mathcal{Q}$ hold). If *ParMol* denotes the number of partial molecules formed, then the precondition of T_2 is the following:

$\texttt{pre}_2 = B' \wedge |Mol \setminus F'| = 0 \wedge ((P(1) \wedge \mathcal{Q}) \vee ParMol = \rho(C))$.

In this task, a relevant issue is that robots have to agree on which molecule μ in F must be formed (in $\rho(C)$ copies).

Definition 10 (Disassembling sequence)**.** *Let F be a pattern and ℓ be a side of $mbr(F)$ encoded with the minimal string within $LSS(F)$. Perform the following iterative process: (1) mark all molecules in F and create an empty ordered list $\mathcal{S}(F)$, (2) with respect to the marked molecules only, compute the set E of all the molecules*

that can be "extracted" from F through ℓ;[3] *(3) insert in $\mathcal{S}(F)$ the molecule $\mu \in E$ with minimum view, (4) unmark all the molecules belonging to the same orbit of μ, (5) iterate from (2) until marked orbits exist. The order of the elements belonging $\mathcal{S}(F)$ constitutes a disassembling sequence for F.*

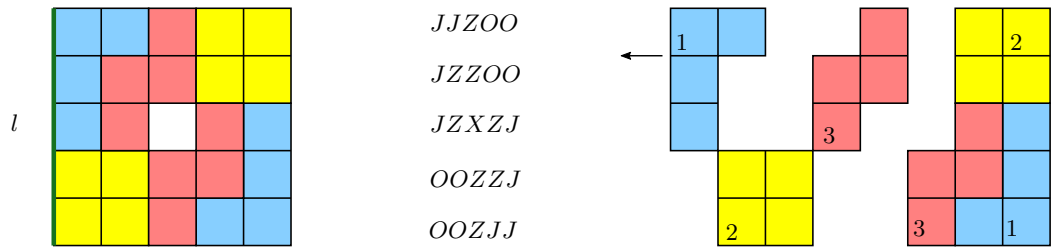


Figure 4.15: From left: a pattern F and the side l; the view of robots; the disassembly sequence. The assembly sequence is OZJ for the first region, ZOJ for the second one.

For instance, $\mathcal{S}(F) = (\mathtt{J}, \mathtt{O}, \mathtt{Z})$ for the pattern F shown in Fig. 4.15 (where ℓ coincides with the left and right sides). The algorithm selects the molecule μ to build by comparing the formed sub-pattern F' with F. According to this comparison, the algorithm searches for molecules μ' and μ'' in F' having minimum and maximum positions in $\mathcal{S}(F)$, respectively; if μ'' is not the last element in $\mathcal{S}(F)$, then μ is the next to μ'' in $\mathcal{S}(F)$, otherwise it precedes μ' in $\mathcal{S}(F)$. In this way, the disassembling sequence in $\mathcal{S}(F)$ is used to correctly compose the pattern.

Let O_1^*, O_2^*, O_3^*, O_4^* be the consecutive orbits closest to $c(R)$ and containing robots not involved in any molecule. Move m_2 works as follows:

- If no partial-molecule is formed, then each robot in O_1^* moves toward the robot in O_2^* belonging to the same region,
- else, each robot closest to $c(R)$, excluding those forming the partial-molecule, moves toward the partial-molecule within the same region toward a position adjacent to the partial-molecule and according to molecule μ to be formed.

The configuration obtained after task T_2 contains $\rho(C)$ **new molecules**.

Tasks T_3 - Central molecule formation. This task processes the configurations of robots fulfilling one among the conditions 2.b, 2.c, and 3.b from Corollary 9 defining the patterns F that are potentially-formable from C.

[3] I.e., when the molecule's projection on ℓ is not obstructed by any other molecule.

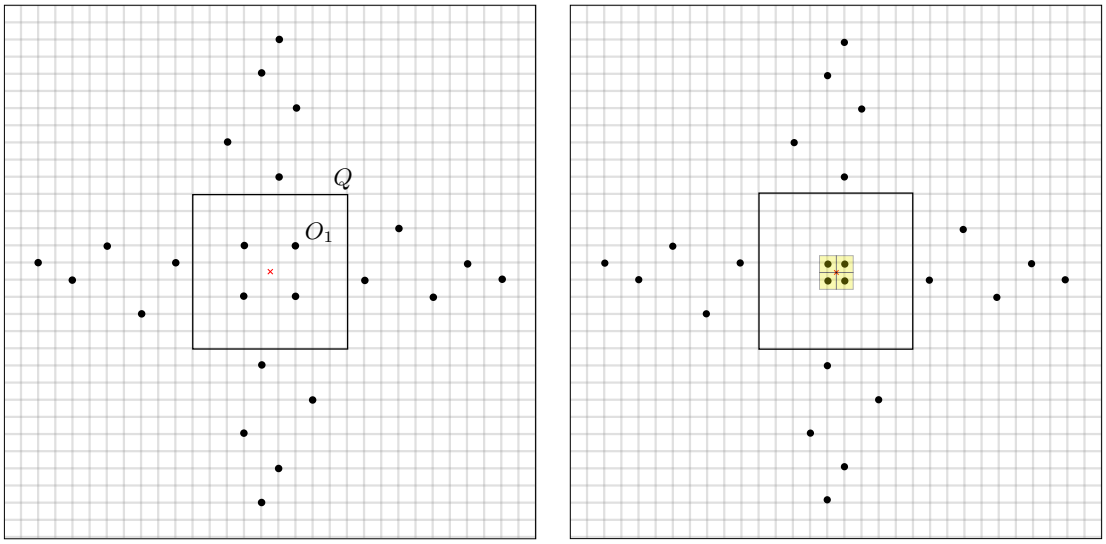


Figure 4.16: Left: A configuration of robots at the end of the task T_1. Consecutive orbits of robots are at distance at least $\delta = 2$ from each other, Q contains at most the orbit of robots closest to $c(R)$. Right: A molecule built in the center of the configuration C.

This task is alternative to T_2 as it builds a single molecule μ in the center of C, usually when it is required to break the symmetry using a molecule. Let B be a Boolean variable that it is true if one among the conditions 2.b, 2.c, and 3.b from Corollary 9 holds. Note that T_3 activates only when the square Q is formed. Let PJR be a boolean variable that is true if there exist two point-joined-robots. In particular, the precondition of T_3 is equal to:

$\texttt{pre}_3 = B \wedge ((P(1) \wedge |Mol| = 0) \vee (P(2) \wedge (ParMol = 1 \vee PJR)))$.

Robots must agree on which molecule in F must be formed as first. It corresponds to the molecule fulfilling conditions 2.*b*, 2.*c*, and 3.*b* of the definition of potentially-formable pattern and with the highest position on the disassembling sequence of F. Four robots belonging to one or two orbits O_1, O_2 (according whether $\rho(C) = 4$ or $\rho(C) = 2$, respectively) closest to $c(R)$ are selected. μ is embedded on the grid so that the center of the molecule coincides with $c(R)$.

Move m_3 works as follows:

- if $\rho(C)$=2, then first robots in O_1 move toward $c(R)$ until reaching the minimum reciprocal distance, then robots in O_2 move toward O_1 by suitably form the desired molecule μ;
- if $\rho(C)$=4, robots in O_1 move toward $c(R)$ until forming $\mu = \texttt{0}$.

The task ends when all the 4 moving robots reach their targets and the molecule is built with its center in $c(R)$. At the end of task T_3, a new configuration C' is obtained with $\rho(C') = 1$.

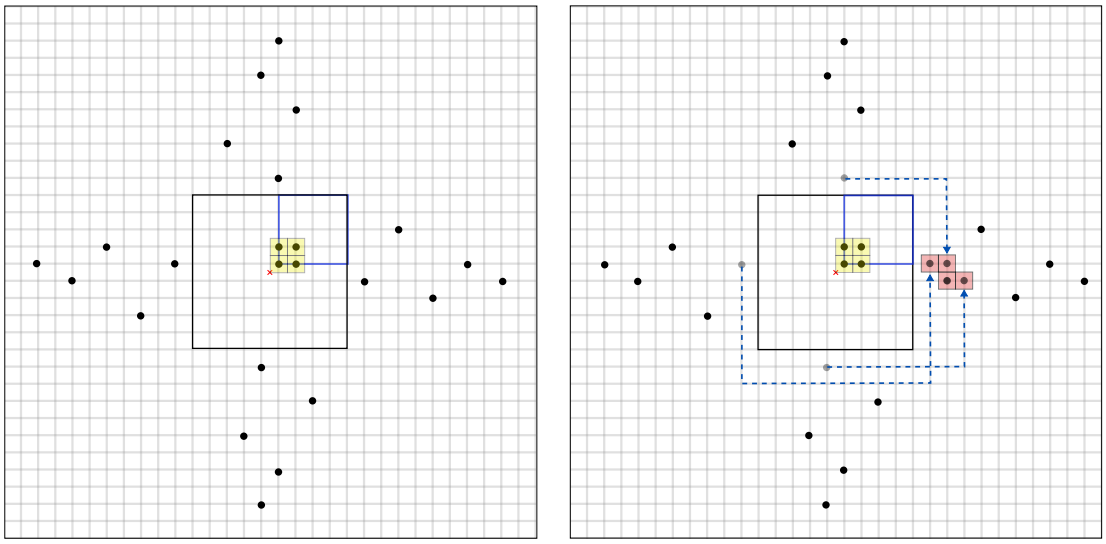


Figure 4.17: Left: a configuration after T_4, the molecule breaks the symmetry by moving away from the center. The blue line shows the embedding of the pattern F. Right: a new molecule is formed.

Task T_4 - Molecules aggregation. During this task, the molecules built during T_2 or T_3 move to start forming F' or to be aggregated to F' (created by previous executions of this task). We define **quadrant** any of the four areas into which the square Q is divided by two orthogonal lines parallel to the sides of Q and intersecting in $c(R)$. To test if the pattern creation has already started, robots check whether (1) there exists a sub-pattern F' in one of the four quadrants, or (2) there exists a sub-pattern F' embedded so that it is centered in $c(R)$. This allows robots to evaluate the following precondition of T_4: $\texttt{pre}_4 = \ |Mol \setminus F'| > 0$.

During this task, $\rho(C)$ can be 1, 2 or 4. To correctly determine the move to be performed, the algorithm considers four disjoint cases, which are defined according to $\rho(C)$, F', and Mol. *(Case 1)* $\rho(C) = 1$, $|F'| = 0$, $Mol = \{\mu\}$, and μ is centered in $c(R)$. It is clear that the current configuration C has been created in T_3. In this case, m_4 breaks the symmetry by simply moving μ away from the center in an arbitrary direction. In the formed configuration C' we have $\rho(C') = 1$, $|F'| = 0$, $Mol = \{\mu\}$, and μ is no longer centered in $c(C')$.

(Case 2) $\rho(C) = 1$, $|F'| = 0$, $Mol = \{\mu\}$, and μ is not centered in $c(R)$. The current configuration has been created in T_4 (Case 1), or by T_2. F is meant to

be embedded in the quadrant q of Q closest to μ, and m_4 moves μ toward the position in the embedded F corresponding to the minimum position of its shape in the disassembling sequence $\mathcal{S}(F)$. Concerning how F is embedded into q: let ℓ be a side of $mbr(F)$ used in the disassembling sequence (cf. Def. 10), and let c be the corner of ℓ with larger label; c is mapped on the vertex in q closest to $c(R)$, and ℓ is mapped on the counter-clockwise internal side of q. This embedding is used whenever the configuration is asymmetric.

(Case 3) $\rho(C) = 1$ and $|F'| > 0$. In this case, there exists only one molecule μ which is not part of F'. Move m_4 moves μ toward its target identified by comparing F' with the position of μ in the disassembling sequence of F.

(Case 4) $\rho(C) > 1$ and $|Mol \setminus F'| = \rho(C)$. In this case, F' is embedded so that it is centered in $c(R)$. There are exactly $\rho(C)$ molecules which are not part of F', and they must be moved toward their final targets. The final targets are obtained by comparing F with F'. During the movements, each molecule remains in the same region. The last time this task is executed, F is finally formed.

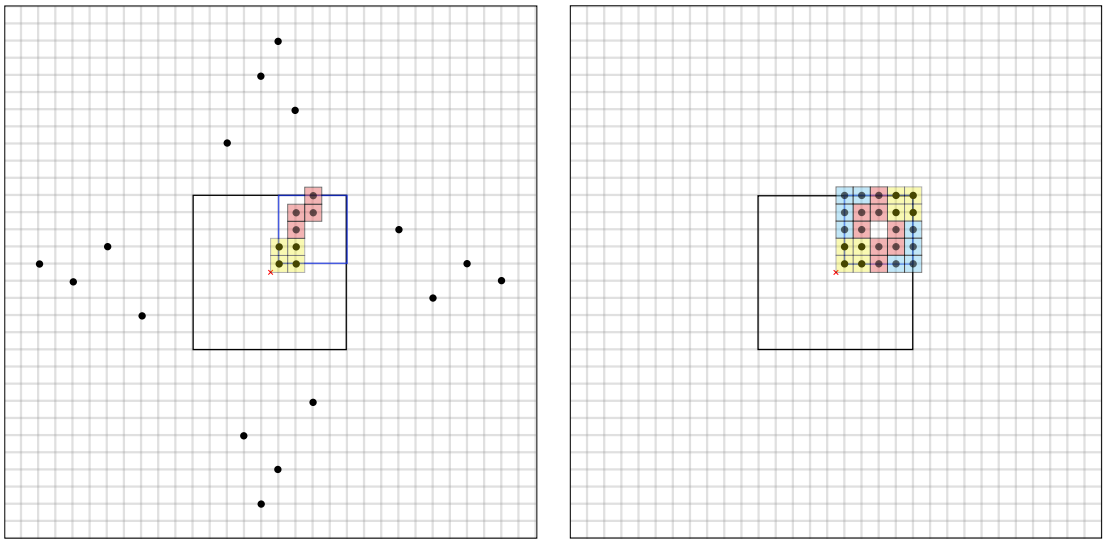


Figure 4.18: Left: the new molecule positioned according to the embedding of F; right: the pattern F is completed.

Task T_5. It refers to the **termination** problem i.e., entities recognize the pattern is formed and no one has to move, $\texttt{pre}_5$ = *"F is formed"*.

Running example

In this section, we show the effectiveness of algorithm $\mathcal{A}_{\mathsf{TL}}$ with an extended example. We recall that $\mathcal{A}_{\mathsf{TL}}$ decomposes the TL-MPF problem into five tasks $T_1, T_2, \ldots, T_5$.

var	*definition*
$P(k)$	each orbit O^i , $i > k$, is correctly positioned wrt $fd_Q()$
$\mathcal{Q}$	square Q is formed with at most one orbit inside
B	one among the conditions 2.b, 2.c, and 3.b of Corollary 9 holds
B'	one among the conditions 1, 2.a, and 3.a of Corollary 9 holds
PJR	there exist two point-joined-robots

Table 4.5: The basic Boolean variables used to define all the tasks' preconditions.

Tables 4.5 and 4.6 summarize all the Boolean variables used to define the tasks' preconditions and all the preconditions, respectively, for such tasks. A predicate $P_i = \texttt{pre}_i \wedge \neg(\texttt{pre}_{i+1} \vee \texttt{pre}_{i+2} \vee \ldots \vee \texttt{pre}_5)$ is assigned to each task T_i. According to this definition of the predicates, $\mathcal{A}_{\mathsf{TL}}$ works as follows: in the `Compute` phase, each entity evaluates - wrt the perceived configuration and the provided input - the preconditions starting from $\texttt{pre}_5$ and proceeding in the reverse order until a true precondition is found.

We now simulate the running of $\mathcal{A}_{\mathsf{TL}}$ by starting from the configuration C_1 shown in Figure 4.14.left. As $\rho(C) = 4$ in C_1, Figure 4.14.right shows the subdivision into four regions and the subset of six robots belonging to one region.

pre	*definition*
$\texttt{pre}_1$	`true`
$\texttt{pre}_2$	$B' \wedge \lvert Mol \setminus F' \rvert = 0 \wedge ((P(1) \wedge \mathcal{Q}) \vee ParMol = \rho(C))$
$\texttt{pre}_3$	$B \wedge ((P(1) \wedge \lvert Mol \rvert = 0) \vee (P(2) \wedge (ParMol = 1 \vee PJR)))$
$\texttt{pre}_4$	$\lvert Mol \setminus F' \rvert > 0$
$\texttt{pre}_5$	F is formed

Table 4.6: Tasks' preconditions.

Since in C there are no molecules formed, then $\texttt{pre}_5$ and $\texttt{pre}_4$ are false. $P(1)$ is false and there is no partial molecule or point-joined robots so $\texttt{pre}_3$ is false. Since B' is false, then $\texttt{pre}_2$ is false. As $\texttt{pre}_1 = \texttt{true}$, then $P_1 = \neg(\texttt{pre}_2 \vee \texttt{pre}_3 \vee \texttt{pre}_4 \vee \texttt{pre}_5)$ and hence It C_1 is in T_1.

Move m_1 is designed to make both $P(1)$ and $\mathcal{Q}$ true. Since $\rho(C) = 4$, m_1 moves all the robots but the 4 closest to $c(R)$ so as to enlarge the $mbr(R)$. This step ensures that robots have enough space to form the designed molecules in the successive tasks. Actually, from C_1, both $P(1)$ and $\mathcal{Q}$ becomes true as soon as configuration

C_2 is achieved, where exactly 4 robots remain inside the square Q (cf. Figure 4.16). When this happens, still no molecules are yet formed, that is $\texttt{pre}_5$, $\texttt{pre}_4$ are false. However $\texttt{pre}_3$ is true: B is true since condition 3.b of Corollary 9 holds and $Mol = 0$, hence C_2 belongs to T_3.

Move m_3 moves 4 robots inside Q to form the first molecule of type `O` in the center of the configuration. Once this happens, C_3 is obtained, $\texttt{pre}_5$ is still false, $|Mol \setminus F'| = 1$, $\texttt{pre}_4$ becomes true and hence the configuration is in T_4, see Figure 4.16.right.

Move m_4 is defined by cases. Since in C_3 we have that $\rho(C) = 1$, $|F'| = 0$, $Mol = \{\mu\}$, and μ is centered in $c(R)$, then Case 1 applies. As a consequence, m_4 breaks the symmetry by moving the molecule away from the center in an arbitrary direction. The obtained configuration is still in T_4, but now Case 2 applies. In fact, $\rho(C_4) = 1$, $|F'| = 0$, $Mol = \{\mu\}$, and μ is not centered in $c(R)$). In this case, F is meant to be embedded in the quadrant of Q closest to μ, and m_4 moves μ toward the position in the embedded F corresponding to the minimum position of its shape in the disassembling sequence $\mathcal{S}(F)$. It is easy to observe that it is sufficient a single LCM cycle to move the molecule in its final destination, as shown in Figure 4.17.left. The obtained configuration is denoted as C_4 and it is still in T_4, but now Case 3 applies.

In C_4, clearly $\texttt{pre}_5$ is false, $|Mol \setminus F'| = 0$ hence $\texttt{pre}_4$ is false. As $\rho(C) = 1$, B is false and so $\texttt{pre}_3$ is, while B' holds since condition 1 of Corollary 9 holds. Both $\mathcal{Q}$ and $P(1)$ hold, so the configuration is in T_2.

Four consecutive orbits of robots are selected, $O_1^*, O_2^*, O_3^*, O_4^*$. Since $\rho(C) = 1$ there is only one robot per orbit. Hence, the robot in O_1^* moves toward the robot in O_2^* belonging to the same region to form a partial molecule and the configuration is still in T_2. Then, the second condition of move m_2 holds, so the robots closest to $c(R)$, excluding those forming the partial-molecule, move toward the partial-molecule within the same region in a position adjacent to the partial-molecule (according to molecule μ to be formed). According to Definition 10, the next molecule to be formed is a `Z`.

Figure 4.15.right shows the assembling and disassembling sequence of the two regions of the pattern F (Figure 4.15.left), according to the minimal string associated with F (Figure 4.15.middle). In particular the assembly sequence of the first region is OZJ because the algorithm was forced to create a molecule `O` as first, whereas the sequence for the second region is ZOJ.

The formation of molecule `Z` is done as shown in Figure 4.17.right, where configuration C_5 is represented.

In C_5, $\texttt{pre}_5$ is false, but $\texttt{pre}_4$ is true. In fact, in T_4 - Case 3 - the algorithm moves the new molecule created by T_2 toward F', positioning it accordingly to the embedding of F. Once this happens, a new configuration belonging to T_2 is obtained, see C_6 in Figure 4.18.left, hence the execution of the algorithm cycles among T_2 and T_4 until pattern F is formed, i.e. configuration C_7, shown in Figure 4.18.right is achieved, where clearly $\texttt{pre}_5$ holds.

4.6.2 Algorithm correctness

In this section, we formally prove that algorithm $\mathcal{A}_{\mathsf{TL}}$ solves the TL-MPF problem. According to the methodology proposed in [44], the correctness of the proposed algorithm can be obtained by proving that all the following properties hold: H_1, H_2, H_3, H_4.

H_1: *The algorithm never generates unsolvable configurations. According to Corollary 9, this implies that This implies that, each configuration $C(t)$, $t > 0$, generated by the algorithm is potentially-solvable.*

H_2: *The movement of each robot is collision-free.*

H_3: *For each task T_i, the transitions from T_i to any other task are exactly those declared in Table 4.4.*

H_4: *Each transition in Table 4.4 occurs after a finite number of cycles. This means that the generated configurations can remain in the same task only for a finite number of cycles.*

Since these properties must be proved for each transition/move, then in the following we provide a specific lemma for each task. Property H_3 does not directly implies that robots/molecules "complete" each task in a finite amount of time. In fact, there is a cycle created by transitions between tasks T_2 and T_4. Anyway, a final theorem will assess the correctness of $\mathcal{A}_{\mathsf{TL}}$ by making use of all the proved properties H_1–H_4 for each task and by also showing that there is a finite number of transitions between tasks T_2 and T_4.

Lemma 24. *Let R be a configuration in T_1. From R, $\mathcal{A}_{\mathsf{TL}}$ eventually leads to a configuration belonging to T_2 or T_3.*

Proof. During task T_1 robots move away from $c(R)$ to leave a square Q of side $2S$ centered in $c(R)$ with at most one orbit inside, while leaving at least δ space between consecutive orbits. Let us analyze properties H_i, for $1 \leq i \leq 4$, separately.

H_1: By move m_1, all the orbits but O^1 move farthest from $c(R)$. During the movements, robots move synchronously keeping the same symmetricity of the initial configuration and the same type of center. The final targets $fd_Q()$ reached by the robots are defined so that each orbit is at a different distance from the center $c(R)$. Due to the synchronicity of the movements, $\rho(C)$ is maintained the same along all the movements and after the robots reach their targets.

H_2: During task T_1, robots increase their distance from $c(R)$ and from other robots moving in a perpendicular direction with respect to the side of $mbr(R)$ to which they are associated with. Orbits move all together and the targets for robots are defined so that the distance between consecutive orbits is at least δ therefore collisions cannot occur during the movements of the robots.

H_3: We show that each configuration generated from C remains in T_1 until all robots reach their targets. When T_1 starts, no molecules are yet formed therefore $\texttt{pre}_5$ is false. During the movements of the robots, no molecules are built so $\texttt{pre}_4$ remains false and the configuration does not belong to T_4. Therefore at the end of T_1 the configuration is either in T_2 or in T_3.

H_4: As long as the configuration remains in T_1, the distance of each moving robot from its target decreases. Hence, the task ends within a finite number of computational cycles.

□

Lemma 25. *Let C be configuration in T_2. From C, $\mathcal{A}_{\mathsf{TL}}$ eventually leads to a configuration belonging to T_4.*

Proof. During task T_2, $\rho(C)$ new molecules are built. Let us analyze properties H_i, for $1 \leq i \leq 4$, separately.

H_1: During this task, the algorithm moves $\rho(C)$ orbits of robots and builds $\rho(C)$ molecules synchronously working by regions. Hence the symmetricity of the configuration is kept during the execution of T_2. If there are only four robots left and they move to form the last molecule, at least two of these four robots are on the sides of $mbr(R)$ holding its shape. When the two outermost robots are on $mbr(R)$, both $mbr(R)$ and the type of center do not change during the movements of O_1^* toward O_2^*. When all the four robots are on $mbr(R)$, then the shape of $mbr(R)$ and the type of center change during the movement of O_1^*. However the type of center changes only either horizontally or vertically, never in both directions so it can never coincide with $tc(F)$, eventually changing $\rho(C)$. So $c(R) \neq c(F)$ during all the execution of T_2 and $\rho(C)$ is kept until the end of the task.

H_2: During task T_2 the four orbits of robots O_1^*, O_2^*,O_3^*,O_4^* that are the closest to $c(R)$ are selected to move. One robot for each region moves at a time. Firstly the robots from O_1^* move towards the robots in O_2^*. Note that, at the end of task T_1 the distance between consecutive orbits is at least δ, so the robots in O_1^* move in an empty space until they form a partial molecule with the robots of orbit O_2^*. The partial molecule is built outside Q. Since O_3^*, O_4^* are the orbits of robots closest to $c(R)$ not involved in any molecule, they move toward the partial molecules without any collision with other robots. Their target is a position adjacent to the partial molecule according to the molecule μ.

H_3: We show that each configuration generated from C remains in T_2 until $\rho(C)$ molecules are built. Since the configuration is in T_2, all the preconditions $\texttt{pre}_i$ with $i > 2$ are false. In fact F is not formed so $\texttt{pre}_5$ is false, and since there

are no molecules that are not part of the pattern until T_2 ends, then $\texttt{pre}_4$ is false. During task T_2, $\rho(C)$ can be 1, 2, or 4. If $\rho(C) = 1$ then one molecule is built and the movements of the robots never augment the symmetricity of the configuration (see Property H_1 above) hence variable B belonging to $\texttt{pre}_3$ is always false. If in T_2, $\rho(C) = 2, 4$ then condition 2 of Corollary 9 is false and the movements of the robots during move m_2 cannot change the symmetricity of the configuration nor the type of center. Therefore variable B' is false and does not change its value during the execution of T_2, so $\texttt{pre}_3$ is false.

H_4**:** As long as the configuration remains in T_2, the distance of each moving robot from their target decreases. Hence, within a finite number of computational cycles the task ends and the configuration is in T_4.

□

Lemma 26. *Let R be a configuration in T_3. From C, $\mathcal{A}_{\mathsf{TL}}$ eventually leads to a configuration belonging to T_4.*

Proof. Let us analyze properties H_i, for $1 \leq i \leq 4$, separately.

H_1**:** During this task $\rho(C) = 2, 4$ and one among conditions 2.b, 2.c, 3.b of Corollary 9 holds. One or two orbits of robots closest to $c(R)$ move. During the movements of $\rho(C)$ robots towards $c(R)$, the symmetricity of C and the type of center cannot change due to the synchronicity of the moves until the task is over. When the task ends, a new configuration C' is obtained such that $\rho(C') = 1$. Condition 1 of Corollary 9 holds and the configuration is still potentially-formable.

H_2**:** Task T_3 starts after T_1, that is square Q centered in $c(R)$ with at most four robots inside has been realized. The four robots closest to $c(R)$ move toward $c(R)$ to build the first molecule. They belong to one or two orbits depending on $\rho(C)$. There are no other robots between these four and $c(R)$, so no collision can occur. As soon as they become adjacent, they stop and the molecule is formed.

H_3**:** The precondition $\texttt{pre}_4$ remains false until the molecule is built. As soon as the molecule is completed, precondition $\texttt{pre}_4$ becomes true and the configuration is in T_4.

H_4**:** As long as the configuration remains in T_3, the distance of each moving robot from $c(R)$ decreases. Hence, within a finite number of computational cycles, the robots create a molecule and the configuration is in T_4.

□

Lemma 27. *Let C be a configuration in T_4. From R, $\mathcal{A}_{\mathsf{TL}}$ eventually leads to a configuration belonging to T_2 or T_5.*

Proof. During this task, the molecules formed during either task T_2 or T_3 move to join the pattern F'. Let us analyze properties H_i, for $1 \leq i \leq 4$, separately.

H_1: If $\rho(C) = 1$, the pattern F is embedded in a quadrant q of Q. One molecule goes toward its target in q and $\rho(C)$ cannot increase during the movement. Then, the configuration remains potentially-solvable by Corollary 9. If $\rho(C) = 2, 4$, during the movements of the molecules, $c(R)$ does not change and so does $tc(C)$, so the conditions 2 and 3 of Corollary 9 still hold.

H_2: During this task only molecules move, therefore collisions between robots cannot happen. When $\rho(C) = 2, 4$, there is one molecule in each region that goes toward F' that is embedded in the center of $c(R)$. The space between the molecules μ and F' is empty and the molecules move on free trajectories. When $\rho(C) = 1$ then one molecule goes toward F' that is embedded in a quadrant q of Q. The quadrant q is big enough to contain F, the space between F' and the molecule μ is empty and the molecule moves on a free trajectory. Therefore no collision can occur. Note that, as more robots are assembled into molecules the empty space around Q enlarges. Moreover the disassembly sequence ensures that molecule can set in place in F without colliding with other molecules.

H_3: Precondition $\mathtt{pre}_4$ remains true until there are molecules that are not yet part of the pattern F'. If there are no robots left then, as soon as the molecules join the pattern F', the pattern F is completed and the configuration is in T_5, otherwise the configuration is in T_2.

H_4: As long as the configuration remains in T_4, the distance of each moving molecule from F' decreases. Hence, within a finite number of computational cycles, the molecules join the pattern.

□

Theorem 10. *$\mathcal{A}_{\mathsf{TL}}$ solves the TL-MPF problem for C and F if and only if F is potentially-formable from C.*

Proof of Theorem 10. Lemmata 24-27 ensure that properties H_1, H_2, H_3, and H_4 hold for each task T_1, $T_2, \ldots, T_5$. Then F is always potentially-solvable; the movements of the robots and molecules are all collision-free; all the transitions are those reported in Table 4.4; and the generated configurations can remain in the same task only for a finite number of cycles. Lemmata 24-27 also show that from a given task only subsequent tasks can be reached, or $\mathtt{pre}_5$ eventually holds (and hence TL-MPF is solved). The only exception is the cycle among tasks T_2 and T_4. However, in this case, at the end of T_4, the number of molecules composing the

pattern increases, and since no molecule is moved away from the pattern, task T_5 is reached from T_4 after a finite number of transitions between T_2 and T_4. This formally implies that, for each initial configuration C and for each execution $\mathbb{E}$: $C = C(t_0), C(t_1), C(t_2), \ldots$ of $\mathcal{A}_{\mathsf{TL}}$, there exists a finite time $t_j > 0$ such that $C(t_j)$ is similar to the pattern to be formed in the TL-MPF problem and $C(t_k) = C(t_j)$ for each time $t_k \geq t_j$.

4.7 Concluding remarks

We proposed $\mathcal{MOBLOT}$, a new theoretical model in the context of the swarm and modular robotics that extends the $\mathcal{OBLOT}$ model. $\mathcal{MOBLOT}$ concerns two levels of computational entities: robots and molecules. Robots can be very weak entities like in the $\mathcal{OBLOT}$ model; molecules are usually more complex entities with an extent. Ideally, robots and molecules are guided by two different distributed algorithms: the former is used to form molecules, the latter to manage molecules, e.g. to assemble them to obtain some complex structure (the matter). Once molecules have accomplished their first task (e.g., the matter is formed), a new task could be further approached by molecules, e.g. rearranging (self-reconfigure) their positions to get a different shape for the matter.

To highlight some potentials of the $\mathcal{MOBLOT}$ model, we have introduced the Matter Formation problem. We have proven the necessary condition to form the matter. According to Theorem 5, the symmetricity of the initial configuration of robots must divide either the symmetricity of a molecule or the symmetricity of the matter to be formed. Interestingly, this implies that the matter could be formed even when the symmetricity of the input configuration is unrelated to that of the matter (in such cases, the molecules play a decisive role). To this respect, we have presented a case study comprising all the conditions of Theorem 5, called HexMF in which the only formable molecule is made of just two robots.

We have considered the $\mathcal{MOBLOT}$ model where robots move along the edges of a graph. We have focused on the Molecular Pattern Formation (MPF) problem where the final configuration is composed only of molecules. For MPF, we have proven a necessary condition for its resolution. As a case study, we have introduced the TL-MPF problem, where robots move along a square grid and the set of formable molecules is the set of the seven tetrominoes. We have identified when TL-MPF is potentially solvable and for all these cases we have provided a resolution distributed algorithm, hence proving a full characterization.

Conclusions

We proposed mathematical models and computational algorithms useful in the context of wildfire management. We presented models and problems for fire preparedness measures. Results come from the joint work with Marc Demange, Gabriele Di Stefano, and Pierpaolo Vittorini. Then, we designed algorithms for the coordination of multi-robotic systems. Results in these topics come from the joint work with Serafino Cicerone, Gabriele Di Stefano, and Alfredo Navarra. This thesis gave the main contributions to the following topics.

Main contributions

Modelling. In Chapter 1, we introduced a graph model able to describe the spread of fire. The model provides a way to compute the probability that an area is set on fire, even if the fire ignited in a different part of the graph. Then, we formulated the Firebreak Location problem to address the optimal location of firebreaks in a landscape to minimize a risk function under budget constraints. Successively, we studied the complexity of the problem on planar graphs.

Algorithms for cases solvable in polynomial time. Due to the hardness of the Firebreak Location problem, we looked for cases solvable in polynomial time and studied the Windy Firebreak Location problem. We presented an efficient polynomial time algorithm on tree topology: given a tree with a subset of vertices on fire, the algorithm outputs the maximum number of vertices that can be saved from a fire.

These results are collected in the paper "A graph theoretical approach to the firebreak locating problem" published by *Theoretical Computer Science journal* [1].

We presented the Infinite Windy Firebreak Location problem, a variation of the Firebreak Location problem defined on infinite graphs. The goal is to identify a subset of edges to remove to avoid burning more than a finite part of the graph. We showed that Infinite Windy Firebreak Location polynomially reduces to the problem of finding a Min Cut in a transportation network for classes of graphs like infinite grid graphs and polyomino-grids, a generalization of grid graphs.

These results are collected in the paper "About the infinite windy firebreak location problem" submitted to journal and currently under review [3].

Heuristics. We also studied a case of the Firebreak Location problem in which all the areas have the same probability to burn. We showed that, when the probability of ignition are equal, the Firebreak Location problem can be reduced to the k-Graph Partition problem that consists in removing a fixed number of edges to split a graph in k connected components of balanced size. Given the hardness of the problem, partitioning the graph in balanced components can be addressed using heuristics. One of the most efficient techniques is multilevel partitioning. We have tested this technique on the geographical area of the North of Corsica and we showed that even partitioning the land into a few parts, leads to a significant reduction of the risk.

These results are collected in the paper "Network theory applied to preparedness problems in wildfire management" published by *Safety Science* journal [2].

Model validation. To show the usability of the graph theoretical model, we presented a case study and applied the model to the landscape of Cap Corse, in the North of Corsica. We described how to compute the extension of the areas, and estimate the probabilities of ignition for each area and the probabilities of spread for each edge. We used open data for the computation of the extension of the areas and data on historical fires, while we used fire simulations to estimate the probabilities of spread.

Risk cartography and web-application. We set up the model and we showed the results on maps. Risk cartography gives an effective visualization of data about wildfire risk. We presented a prototype web application designed to offer to target end-users, wildfire, risk managers, and fire agencies an easy-to-use tool. Users can interact with risk cartography easily without knowing the technical details of the underlying data or how algorithms have been implemented. Users can then concentrate on the design of preparedness strategies to see their effect before deployment.

These results were presented to the conference *Fire Ecology across boundaries: connecting science and management*, a paper is under preparation.

Robot formation. In Chapter 3, we proposed a solution for the arbitrary pattern formation problem in which robots, must be able to organize according to any geometric shape given in input.

These results appeared in the proceedings of the conference *International Conference on Distributed Computing and Networking (ICDCN)* in 2021 [4] and the extended paper with the title "Arbitrary pattern formation on infinite regular tessellation graphs" was published by *Theoretical computer Science* journal [7].

Mutual Visibility on trees. We studied the mutual visibility problem for robots moving on graphs, called Geodesic Mutual Visibility problem (GMV). This problem asks to place robots so that they are geodesic mutually visible: each couple of robots has a shortest path in which no other robot resides. The study is motivated by observing that mutually visible robots can reach any other robot along the shortest path without collision. We have proposed a deterministic and distributed algorithm to solve GMV on trees.

These results are collected in the paper "The Geodesic Mutual Visibility Problem for Oblivious Robots: the case of Trees", published in the proceedings of the conference *International Conference on Distributed Computing and Networking (ICDCN)* 2023 [8].

The Moblot model. In Chapter 4 we introduced $\mathcal{MOBLOT}$ a novel model in theoretical swarm robotics in which robots cluster to form bigger computational units, called molecular robots, inspired by the chemical paradigm in which atoms combine to make molecules. Once bonded, molecular robots move in a coordinated way as a new macro entity. $\mathcal{MOBLOT}$ allows us to model a swarm of robots divided into subgroups and modular robotics. Furthermore, we presented the matter formation problem in which robots first cluster into molecular robots, and then molecular robots move to form a pattern introducing a hierarchical and modular approach to solve the pattern formation problem. We have shown that $\mathcal{MOBLOT}$ extends the well-known $\mathcal{OBLOT}$ model [73] and that molecular robots can break certain symmetries that are not solvable in $\mathcal{OBLOT}$. We presented a case study for a particular case of matter formation for robots moving on the Euclidean Plane to illustrate the extended capabilities of the model with respect to $\mathcal{OBLOT}$.

These results are collected in the paper "MOBLOT: molecular oblivious robots" published in the proceedings of the conference *Autonomous Agents and Multiagent Systems (AAMAS)* 2021 [5]. The extended version of the paper has been submitted to journal and it is currently under review.

Moblot on grids. We applied the $\mathcal{MOBLOT}$ model to robots moving on square grid graphs. Grids can be interpreted as a discretization of the plane in which the movements of the robots are quantized and the system has a unit of measurement. We modeled the set of formable molecular robots with polyominoes. We presented and characterized the Tetris-like matter formation problem in which the set of formable molecules is the set of the seven possible tetrominoes.

These results are collected in the paper " Molecular robots with chirality on grids" published in the proceedings of the conference *International Symposium on Algorithms and Experiments for Wireless Sensor Networks (ALGOSENSORS)* 2022 [6].

Future work

We addressed problems arising from the context of wildfire management from a theoretical point of view. Preparedness problems must meet multiple constraints and the coordination of multi-robot systems is algorithmically challenging. Research on these topics brings out new questions, opening new possible future directions.

In the first part of the thesis, we studied the FIREBREAK LOCATION problem and proved its hardness. We presented cases solvable in polynomial time and some heuristic solutions. The hardness results motivate further studies to identify new approximations and new classes of instances solvable in polynomial time. We have shown how to apply the model to a practical case study. It would be interesting to collaborate with fire agencies to test the effectiveness of the risk maps in improving the decision process in fire preparedness interventions. To this aim, the use of the prototype application would help in evaluating the impact of the prevention actions before deployment. Moreover, we must involve stakeholders and administrative people in the estimation of variables of the model like the value for each watershed and the cost of each firebreak installation. Indeed, the value estimation must take into account different points of view and characteristics of the land like the destination of use (urbanized, cultivated, or protected). Similarly, the estimation of a firebreak cost must take into account the environmental impact, the orography, the accessibility of the territory, and the type of fuel. Furthermore, the estimation of the probabilities of ignition could be based not only on past fires, as these data are not always available but also on the updated status of the fuel type. Cover land maps classify the materials covering the land surface using Earth observation satellites. The continuous monitoring from the satellites allows the updating of maps over time to detect the changes in fuel type. The sizing of watersheds could also, be adapted to the risk level, allowing the partitioning of areas at higher risks in smaller watersheds. The simulations would also be enhanced with real-time meteorological data.

In the second part of the thesis, we studied two cases of pattern formation problems under the $\mathcal{OBLOT}$ model. Firstly we addressed the arbitrary pattern formation problem(APF) for robots moving on tessellation graphs and starting from an asymmetric configuration. As a relevant improvement, our algorithm works for any tessellation graph and admits patterns with multiplicities. This work can be extended to accept as input also leader configurations. On the Euclidean plane, APF can be solved if and only if the initial configuration is a leader configuration. Leaders can be moved to break the symmetry. However, in grids, movements are restricted to neighbors and this space discretization poses new challenges. The problem of symmetry breaking on triangular and square grids has been recently solved in [33]. It would be interesting to check whether the algorithm developed in [33] can be combined with our $\mathcal{A}_{form}$ algorithm. If possible, this would characterize the APF problem on both triangular and square grids. It would be also interesting to

investigate the possibility to form a pattern F with multiplicities when robots do not perceive multiplicities. Our algorithm uses multiplicity detection only in the Finalization phase when the two last robots complete the pattern.

Regarding the Geodesic Mutual Visibility problem (GMV), the first open question is the full characterization within the SSync scheduler. The main difficulty arising within any of the available schedulers comes from the management of critical vertices. We have provided some hints about possible strategies, but it is challenging to find a general one. As we have seen, this is especially evident within FSync. It is also interesting to study the time complexity of the resolution algorithms because of the gap between the lower bound and the complexity of the proposed algorithm. As a wider research area, the geodesic mutual visibility problem could be studied on other graph topologies or general graphs. Finally, the study of GMV in asymmetric graphs or grid graphs deserves main attention, even in the Async case.

We introduced $\mathcal{MOBLOT}$, a new model for swarm and modular robotics for the coordination of robots clustered in groups, called molecular robots. We introduced the Matter Formation problem that solves fundamental robot problems like pattern formation in a modular and hierarchical way. There are many directions for the $\mathcal{MOBLOT}$ model. The most natural one is to study a complete characterization of the general Matter Formation problem. Further studies could address the reconfigurability of the matter and, at a higher hierarchical level, the matter movement.

Appendix A

Complexity of Windy Firebreak location in planar instances

In this section, we investigate the complexity of Windy Firebreak Location in restricted planar instances that are natural in a real context. In Section A.1, we first study a restricted version of Planar Max 2SAT that is used for our main reduction. We define the problem and prove its NP-completeness (Proposition A.1.9 and Corollary A.1.10). Then, in Section A.2, we prove that Windy Firebreak Location is NP-complete in bipartite planar graphs of degree at most 5 in the polynomially bounded case, i.e., with vertex values and edge costs bounded by a polynomial function (Proposition A.2.8 and Corollary A.2.9). We use a reduction from the restricted version of Planar Max 2SAT introduced in the previous section. Finally, in Section A.3, we use self refinements to show that the problem remains NP-complete in bipartite planar graphs of degree at most 4 and with all values of vertices and costs of edges equal to 1 (Theorem 1.2.2 that is the main result of Section A). We conclude the section with a refinement in grid graphs with binary vertex values and edge costs (Proposition 1.2.3).

A.1 A restricted version of Planar Max 2SAT

In this section, we study Restricted Strong Planar Max 2SAT, a restricted version of Planar Max 2SAT. We define the problem in Paragraph A.1.1. Then, we prove that Restricted Strong Planar Max 2SAT is NP-complete using a reduction from a restricted version of Planar 3SAT. We present this starting problem in Paragraph A.1.2 and present the reduction step-by-step in Paragraphs A.1.3, A.1.4, A.1.5 and A.1.6. The main result is explained in Paragraph A.1.7.

A.1.1 Definition of Restricted Strong Planar Max 2SAT

To establish Proposition A.1.9 and Corollary A.1.10, the main results of Section A.1, we need a restricted version of Max 2SAT, the Maximum 2-Satisfiability problem (see [77]).

A SAT instance Φ is defined as a set X of n Boolean variables and a set C of m clauses, each defined as a set of literals: every variable x corresponds to two literals x (the positive form) and $\bar{x}$ (the negative form). A k-clause, $k \geq 1$, is a clause containing exactly k literals, all different. To simplify the notation we denote by $(\ell_1, \ldots, \ell_k)$ the k-clause with literals $\ell_i, i = 1, \ldots k$, without distinction between the orders in which they are listed. A truth assignment assigns a Boolean value (True or False) to each variable corresponding to a truth assignment of opposite values for the two literals x and $\bar{x}$: $\bar{x}$ is True if and only if x is False. For a literal $\ell \in \{x, \bar{x}\}$ we denote by $\bar{\ell}$ its negation: $\bar{\ell} = \bar{x}$ if $\ell = x$ and $\bar{\ell} = x$ if $\ell = \bar{x}$. A clause is satisfied if at least one of its literals is satisfied. The usual SAT problem asks whether there is a truth assignment satisfying all clauses and 3SAT is the restriction where Φ contains only 3-clauses. In what follows, we assume that no clause contains two opposite literals (x and $\bar{x}$), in which case the clause would be always satisfied.

A Max 2SAT instance is a SAT instance Φ with only 2-clauses, but this time the aim is to find a truth assignment on X maximizing the number of clauses that are satisfied.

In planar versions of SAT problems one usually considers the bipartite graph $G_\Phi = (X \cup C, E)$ with an edge $xc \in E$ between a variable vertex $x \in X$ and a clause vertex $c \in C$ if either x or $\bar{x}$ appears in the clause c. Planar Max 2SAT is the restricted version of Max 2SAT when the graph G_Φ is planar. An edge xc of G_Φ can be labeled either with the literal x if $x \in c$ or $\bar{x}$ if $\bar{x} \in c$.

Here, we need a restricted condition of planarity, called *strong planarity*, ensuring that there is a planar embedding of G_Φ such that, for every variable vertex x, all the edges incident to x and with the same label can be "grouped" on the same side given an orientation of the 2 dimensional plane. More formally we use the graph $\widetilde{G}_\Phi$ obtained from G_Φ by replacing variable vertices by a path P_3 as follows:

- Every variable x is associated with a path P_3 $xx'\bar{x}$ with vertices $x, x', \bar{x}$ and edges $xx', x'\bar{x}$.
- Every clause c is associated with a vertex c.
- There is an edge cx (respectively $c\bar{x}$) if the literal x (respectively $\bar{x}$) appears in the clause c.

A SAT instance Φ will be called *strongly planar* if $\widetilde{G}_\Phi$ is planar.

Note that $\widetilde{G}_\Phi$ is still bipartite; moreover if $\widetilde{G}_\Phi$ is planar, then G_Φ is also planar

but the converse is not true. Consider for instance the instance Φ_0 with variables $X_0 = \{x, a, b, c, d\}$ and clauses $C_0 = \{(x, a), (x, c), (\bar{x}, b), (\bar{x}, d), (a, b), (b, c), (c, d), (a, d)\}$.$G_{\Phi_0}$ is planar since it is a subdivision of the graph obtained from a C_4 *abcd* by adding a universal vertex x (a wheel on five vertices). However, $\widetilde{G}_{\Phi_0}$ is not planar since it contains a subdivision of the complete bipartite graph $K_{3,3}$ which is not planar.

STRONG PLANAR MAX 2SAT is defined as the restriction of MAX 2SAT for which $\widetilde{G}_\Phi$ is planar; it is a restricted case of PLANAR MAX 2SAT. We then consider the RESTRICTED STRONG PLANAR MAX 2SAT defined as follows (decision version):

Definition A.1.1. RESTRICTED STRONG PLANAR MAX 2SAT
Instance: a MAX 2SAT *instance $\Phi = (X, C)$ defined by a set of boolean variables X and a set of 2-clauses C as well as an integer $K \leq |C|$ with the following restrictions:*
(i): $\widetilde{G}_\Phi$ is planar;
(ii): each literal appears in at most 4 clauses (So, $\widetilde{G}_\Phi$ is of maximum degree 5).
Question: is there a truth assignment of variables such that at least K clauses are satisfied?
We denote such an instance by (Φ, K) or (X, C, K).

The decision version of MAX 2SAT is known to be NP-complete [78] and moreover, the reduction preserves planarity [84], thus inducing that the decision version of PLANAR MAX 2SAT is NP-complete using PLANAR 3SAT [101]. Unfortunately, the reduction devised by Garey et al. [78] does not preserve the strong planarity property and we need to slightly modify and rewrite the argument given by Guibas et al. [84]. To this aim, we devise a polynomial reduction from the following restricted version of PLANAR 3SAT.

A.1.2 The starting problem: a restricted version of PLANAR 3SAT

The following problem is known as NP-complete [49]:

Definition A.1.2. RESTRICTED PLANAR 3SAT*:*
Instance Φ: a set X of boolean variables and a set C of 2-clauses and 3-clauses. The graph G_Φ is planar and every variable x appears in exactly two 2-clauses and one 3-clause. Moreover, x appears once in positive form (literal x) and once in negative form (literal $\bar{x}$) in the 2-clauses.
Question: is there a truth assignment to variables that satisfies all clauses?

Consider an instance $\Phi = (X, C)$ of RESTRICTED PLANAR 3SAT (Definition A.1.2). Denote by $C = C_2 \cup C_3$ where C_2 is the set of 2-clauses and C_3 is the set of 3-clauses. Since every variable appears in exactly three clauses, if G_Φ is planar, so does $\widetilde{G}_\Phi$. We describe below how to build from $\Phi = (X, C)$ an instance $(\Phi^*, K) = (X^*, C^*, K)$

of Restricted Strong Planar Max 2SAT (Definition A.1.1) such that Φ is satisfiable if and only if (Φ^*, K) is satisfiable.

In the transformation, 2-clauses in C_2 remain unchanged. In Paragraph A.1.3 we describe how to transform each 3-clause in C_3 with a collection of 14 2-clauses and we highlight the main properties of this gadget. In Paragraph A.1.4, we describe the whole instance Φ^*. Then, in Paragraph A.1.5 we use the main property of the 3-clause gadget to ensure that the reduction is correct. Finally, in Paragraph A.1.6 we justify that (Φ^*, K) is an instance of Restricted Strong Planar Max 2SAT.

A.1.3 Case of 3-clauses

For every 3-clause $c = (\ell_1, \ell_2, \ell_3)$, where ℓ_i, $i = 1, 2, 3$ are literals associated with variables in X, we proceed in two phases. We first introduce a new variable a_c and replace c with six 2-clauses and four 1-clauses : c is replaced by the set of ten clauses $\mathcal{C}'_c = \{(\ell_1, \ell_2), (\ell_1, \ell_3), (\ell_2, \ell_3), (\ell_1, a_c), (\ell_2, a_c), (\ell_3, a_c), (\bar{\ell}_1), (\bar{\ell}_2), (\bar{\ell}_3), (\bar{a}_c)\}$. We use this set of clauses in Claims A.1.3 and A.1.4 since it makes these claims easier. Then, for any of these 1-clauses $(\ell) \in \mathcal{C}'_c$, where ℓ is a literal corresponding to a variable in X, we add a new variable r_c and we replace the clause (ℓ) with the set of two 2-clauses $C^{\ell}_c = \{(\ell, r_c), (\ell, \bar{r}_c)\}$.

We then denote $\mathcal{C}_c = \{(\ell_1, \ell_2), (\ell_1, \ell_3), (\ell_2, \ell_3), (\ell_1, a_c), (\ell_2, a_c), (\ell_3, a_c), (\bar{\ell}_1, r^1_c), (\bar{\ell}_1, \bar{r}^1_c), (\bar{\ell}_2, r^2_c), (\bar{\ell}_2, \bar{r}^2_c), (\bar{\ell}_3, r^3_c), (\bar{\ell}_3, \bar{r}^3_c), (\bar{a}_c, r^a_c), (\bar{a}_c, \bar{r}^a_c)\}$ the set obtained from $\mathcal{C}'_c$ by replacing the 1-clauses with the related pair of 2-clauses with the new r-variables. We denote R the set of all r-variables added in the treatment of 3-clauses.

We then emphasize the following properties that are useful in the reduction:

Claim A.1.3. *For every $c \in C_3$, at most seven clauses in $\mathcal{C}'_c$ can be simultaneously satisfied.*

Proof. Suppose a_c is False, then at most three clauses among $(\ell_1, a_c), (\ell_2, a_c), (\ell_3, a_c)$, $(\bar{\ell}_1)$, $(\bar{\ell}_2)$ and $(\bar{\ell}_3)$ can be True simultaneously, thus at least three clauses are False. If a_c is True and at least two literals among ℓ_1, ℓ_2, ℓ_3 are True, then at least two clauses among $(\bar{\ell}_1), (\bar{\ell}_2), (\bar{\ell}_3)$ are False; with $(\bar{a}_c)$, it makes at least three unsatisfied clauses. Finally, if a_c is True and at least two literals among ℓ_1, ℓ_2, ℓ_3 - say without loss of generality ℓ_1, ℓ_2 - are False, then the clause (ℓ_1, ℓ_2) is False and the three remaining clauses $(\ell_1, \ell_3), (\ell_2, \ell_3), (\bar{\ell}_3)$ cannot be simultaneously satisfied, leaving at least three unsatisfied clauses in $\mathcal{C}'_c$. □

Claim A.1.4. *For every $c \in C_3$, given any truth assignment of literals ℓ_1, ℓ_2, ℓ_3, if c is not satisfied then at most six clauses in $\mathcal{C}'_c$ can be simultaneously satisfied. If c is satisfied, then there is a truth assignment of variable a_c such that seven clauses in $\mathcal{C}'_c$ are satisfied.*

Proof. Suppose c is not satisfied, then either a_c is True and the four clauses (ℓ_1, ℓ_2), (ℓ_1, ℓ_3), (ℓ_2, ℓ_3), $(\bar{a}_c)$ are unsatisfied or a_c is False and the six clauses (ℓ_1, ℓ_2), (ℓ_1, ℓ_3),

(ℓ_2, ℓ_3), (ℓ_1, a_c), (ℓ_2, a_c), (ℓ_3, a_c) are unsatisfied. Suppose now one literal - say ℓ_1 - is True while ℓ_2, ℓ_3 are False, then choosing a_c True leaves only three unsatisfied clauses $(\bar{\ell_1}), (\ell_2, \ell_3)$ and $(\bar{a}_c)$. Suppose now that ℓ_1, ℓ_2 are True while ℓ_3 is False, then any truth value for a_c leaves only three unsatisfied clauses, $(\bar{\ell_1}), (\bar{\ell_2})$ and either (ℓ_3, a_c) or $(\bar{a}_c)$. Finally if the three literals ℓ_1, ℓ_2, ℓ_3 are True, then choosing a_c False leaves only three unsatisfied clauses, $(\bar{\ell_1}), (\bar{\ell_2})$ and $(\bar{\ell_3})$.

All other configurations are symmetrical. □

Finally, we have:

Claim A.1.5. *For any 1-clause $(\ell) \in \mathcal{C}'_c$, if ℓ is True, then the two 2-clauses in C_c^ℓ are satisfied while if ℓ is False, then any truth assignment for r_c satisfies only one of the two clauses in C_c^ℓ.*

This last claim means that, when replacing 1-clauses in $\mathcal{C}'_c$ with two 2-clauses, the number of true clauses increases by 4 for any truth assignment.

As a consequence of Claims A.1.3, A.1.4, and A.1.5, any satisfied clause in C_3 generates 7+4=11 satisfied clauses in the new instance while unsatisfied clauses in C generate at most 6+4=10 satisfied clauses in the new instance.

A.1.4 The instance (Φ^*, K)

Putting all together, if we denote by X^* the new set of variables and by C^* the new set of clauses we have:

$$\begin{aligned} X^* &= X \cup \{a_c \mid c \in C_3\} \cup R \\ C^* &= C_2 \cup \bigcup_{c \in C_3} \mathcal{C}_c. \end{aligned} \tag{A.1}$$

To finalize the reduction, we set $K = |C_2| + 11|C_3| = |C| + 10|C_3|$. X^*, C^*, K define the instance (Φ^*, K).

Equation A.1 immediately justifies the following claim:

Claim A.1.6. *The construction from (X, C) to (X^*, C^*, K) can be performed in polynomial time.*

In the next paragraph, we show that (Φ^*, K) is satisfied if and only if the instance Φ is satisfied.

A.1.5 Validity of the reduction

The above discussion immediately shows:

Claim A.1.7. *There is a truth assignment of variables in X satisfying all clauses in C if and only if there is a truth assignment of variables in X^* satisfying at least K clauses in C^*.*

Proof. Note that Claims A.1.3 and A.1.5 ensure that at most K clauses can be simultaneously satisfied in (X^*, C^*).

Assume first there is a truth assignment of variables in X satisfying all clauses in C. Claims A.1.3, A.1.4 and A.1.5 show that for each satisfied 3-clause $c \in C_3$ there is a truth assignment for the related variable a_c inducing 11 satisfied clauses in $\mathcal{C}_c$, for any truth assignment of variables in R. In all it makes $|C_2| + 11|C_3| = K$ satisfied clauses in C^*.

Conversely, consider a truth assignment t of variables in X^* and consider the induced truth assignment t' for variables in $X \subset X^*$. If one clause $c_0 \in C$ is not satisfied, then Claims A.1.3, A.1.4 and A.1.5 ensure that at most $(|C| - 1) + 10|C_3| = K - 1$ clauses of C^* are satisfied. So, if t satisfies at least K clauses in C^*, then all clauses in C are satisfied by the truth assignment t'. □

Finally, we conclude the proof in the next paragraph by showing that $(X^*, C^*, K) = (\Phi^*, K)$ satisfies all properties of RESTRICTED STRONG PLANAR MAX 2SAT instances (see Definition A.1.1).

A.1.6 Properties of the constructed instance

In what follows, for notations related to the reduction, the reader is referred to Paragraphs A.1.3 and A.1.4. This paragraph is dedicated to the following claim:

Claim A.1.8. $(X^*, C^*, K) = (\Phi^*, K)$ *is an instance of* RESTRICTED STRONG PLANAR MAX 2SAT.

Proof. Let us justify that every literal in Φ^* appears in at most four clauses. Consider first a variable $x \in X$. In C, each of the two relative literals x and $\bar{x}$ appears in one 2-clause in C_2: $c_{2,x}$ and $c_{2,\bar{x}}$, respectively, and either x or $\bar{x}$ - say without loss of generality x - appears in one 3-clause $c_{3,x} \in C_3$. Then, in C^*, x appears in the clause $c_{2,x}$ as well as in three 2-clauses in $C_{c_{3,x}}$ while $\bar{x}$ appears in three clauses of C^*: the clause $c_{2,\bar{x}}$ as well as two clauses $(\bar{x}, r), (\bar{x}, \bar{r})$ with $r \in R$. Consider now a variable $a_c, c \in C_3$: the literal a_c appears in three 2-clauses of C_c while $\bar{a}_c$ appears in two clauses $(\bar{a}_c, r), (\bar{a}_c, \bar{r})$ with $r \in R$.

Finally, each literal $r, \bar{r}$ corresponding to variables $r \in R$ appears in a single 2-clause in C^*.

To conclude the proof, note that $\widetilde{G}_{\Phi^*}$ is of maximum degree 5 and we need to show that it is planar. As mentioned previously, $\widetilde{G}_{\Phi}$ is planar since every variable appears in three clauses in Φ. As described in Paragraphs A.1.3 and A.1.4, $\widetilde{G}_{\Phi^*}$ is obtained from $\widetilde{G}_{\Phi}$ in three steps, each preserving planarity.

First, for any 3-clause $c = (\ell_1, \ell_2, \ell_3)$ in C, assume without loss of generality that the literal ℓ_1 appears in the 2-clause (ℓ_1, ℓ_4) while $\bar{\ell}_1$ appears in the 2-clause $(\bar{\ell}_1, \ell_5)$. $\widetilde{G}_{\Phi}$ has been modified as follows. Remove from $\widetilde{G}_{\Phi}$ the clause-vertex (ℓ_1, ℓ_2, ℓ_3) and its three incident edges towards the literal-vertices ℓ_1, ℓ_2 and ℓ_3; add a path $a_c a'_c \bar{a}_c$, new clause vertices (a_c, ℓ_1), (a_c, ℓ_2), (a_c, ℓ_3) and the six edges $a_c(a_c, \ell_i), (a_c, \ell_i)\ell_i$, $i = 1, 2, 3$. Add the three clause-vertices $(\ell_1, \ell_2), (\ell_1, \ell_3)$ and (ℓ_2, ℓ_3), respectively and add the six edges $(\ell_i, \ell_j)\ell_i, (\ell_i, \ell_j)\ell_j, 1 \leq i < j \leq 3$. Then, consider a 1-clauses (ℓ) added for the clause $c = (\ell_1, \ell_2, \ell_3)$: $\ell \in \{\bar{\ell}_1, \bar{\ell}_2, \bar{\ell}_3, \bar{a}_c\}$. Denote by $r^i_c, i = 1, 2, 3$ the r-variable associated with the 1-clause $(\bar{\ell}_i), i = 1, 2, 3$ and r^a_c the r-variable associated with the 1-clause $(\bar{a}_c)$. Add in $\widetilde{G}_{\Phi}$ a cycle on 6 vertices for each of these 1-clauses:, $\bar{\ell}_i - (\bar{\ell}_i, r^i_c) - r^i_c - r'^i_c - \bar{r}^i_c - (\bar{\ell}_i, \bar{r}^i_c) - \bar{\ell}_i$ for $i = 1, 2, 3$ and $\bar{a}_c - (\bar{a}_c, r^a_c) - r^a_c$- $- r'^a_c - \bar{r}^a_c - (\bar{a}_c, \bar{r}^a_c) - \bar{a}_c$. As illustrated in the figure, this transformation preserves planarity since the added gadget is planar and all dashed edges linking it to the rest of the graph can be preserved without crossing other edges.

Since all these two transformations preserve planarity and since $\widetilde{G}_{\Phi}$ is planar, $\widetilde{G}_{\Phi^*}$ is planar, which completes the proof of Claim A.1.8. □

A.1.7 The main result

Finally, we put all together in this paragraph to prove the following proposition:

Proposition A.1.9. Restricted Planar 3SAT *polynomially reduces to* Restricted Strong Planar Max 2SAT.

Proof. The reduction is described in Paragraphs A.1.3 and A.1.4. Claim A.1.6 ensures it is polynomial. Claim A.1.8 justifies that the constructed instance is an instance of Restricted Strong Planar Max 2SAT and Claim A.1.7 justifies that is a reduction from Restricted Planar 3SAT to Restricted Strong Planar Max 2SAT. □

We immediately deduce:

Corollary A.1.10. Restricted Strong Planar Max 2SAT *is NP-complete.*

Proof. Using Proposition A.1.9 and the fact that Restricted Planar 3SAT (Definition A.1.2) is NP-complete [49], it remains to prove that Restricted Strong Planar Max 2SAT is in NP. Given a truth assignment, one can decide in $O(m+n)$ whether it satisfies at least K clauses. This concludes the proof. □

A.2 Hardness of Windy Firebreak location in planar graphs: the case of polynomially bounded edge and vertex weights

In the rest of Section A, we consider Windy Firebreak Location. We recall it is defined on a mixed graph and corresponds to the case where all probabilities of spread are 1. Since Windy Firebreak Location is a particular case of Firebreak Location, all hardness results for the former apply to the latter. In Paragraph A.2.1 some preliminary remarks are provided. The remaining paragraphs are devoted to prove that Windy Firebreak Location is NP-complete in bipartite planar graphs with **polynomially bounded edge costs and vertex values**, using a reduction from Restricted Strong Planar Max 2SAT. Paragraph A.2.2 shows a property of the main gadget used in the proof, Paragraph A.2.3 provides the details of the reduction, Paragraph A.2.4 explains how to choose the values of some parameters needed in the reduction and Paragraph A.2.5 concludes the argument. The main result is stated in the last Paragraph A.2.6.

A.2.1 Preliminary remarks

Firstly note that the computation of ρ is in general $\sharp P$-hard [30], but $\rho(G_H)$ is given in Proposition 1.1.3 when the probability of spread is set to 1 for each edge. Clearly, this calculation can be performed in polynomial time. This means that given a cut system H we can check in polynomial time if it is a solution. As a consequence, contrary to Firebreak Location, Windy Firebreak Location is in NP.

Lemma A.2.1. Windy Firebreak Location *is in NP.*

Let us also note that, if we allow any value for edge costs and vertex values, then a simple reduction from Partition shows that Windy Firebreak Location is NP-complete on stars. We recall that an instance of Partition is a list of n numbers $s_1, \ldots, s_n$ with $\sum_{i=1}^{n} s_i = 2S$ for an integer S. The question is whether there is a subset $A \subset \{1, \ldots, n\}$ such that $\sum_{i \in A} s_i = S$. Partition is known to be NP-complete [77].

Proposition 1.2.1. Partition *polynomially reduces to* Windy Firebreak Location *on stars.*

Proof. Consider an instance of Partition consisting of n integers $s_1, \ldots, s_n$ with $\sum_{i=1}^{n} s_i = 2S$ for an integer S.
We build an instance of Windy Firebreak Location as follows: consider, as G, a star with a center o and n leaves $\ell_1, \ldots, \ell_n$. The values are defined as follows: $\varphi(o) = 0$ and $\varphi(\ell_i) = s_i, i = 1, \ldots, n$ and the costs of edges are given by $\kappa(o\ell_i) =$

$s_i, i = 1, \ldots, n$. Finally o has probability of ignition 1 and other vertices have a probability of ignition 0. We choose $B = R = S$. The construction can be performed in polynomial time.

With this set-up, a cut system H is defined as $H = \{o\ell_i, i \in A \subset \{1, \ldots, n\}\}$, $\kappa(H) = \sum_{i \in A} s_i$ and $\rho(G_H) = \sum_{i \notin A} s_i$. This instance of Windy Firebreak Location is a yes-instance if and only if there is a set $A \subset \{1, \ldots, n\}$ such that $\sum_{i \in A} s_i \leq S = R$ and $\sum_{i \notin A} s_i \leq S = R$. Since $\sum_{i \notin A} s_i + \sum_{i \in A} s_i = 2S$, the instance of Windy Firebreak Location is a yes-instance if and only if the instance of Partition in a yes-instance. This completes the proof. □

We immediately deduce from Lemma A.2.1 and that Partition is NP-complete [77]:

Corollary A.2.2. Windy Firebreak Location *is NP-complete on stars.*

Since Partition is weakly NP-complete, the previous reduction does not give any information about complexity when edge costs and vertex values are polynomially bounded. We focus on this case for the Windy Firebreak Location problem. The rest of Section A.2 aims to prove Proposition A.2.8 and Corollary A.2.9 that establish that Windy Firebreak Location is NP-complete in bipartite planar graphs with polynomially bounded edge costs and vertex values. The proof is based on a polynomial time reduction from Restricted Strong Planar Max 2SAT.

A.2.2 A useful property for the main gadget

Here, we outline a technical lemma that is useful for the reduction.

Lemma A.2.3. *Let $G = (V, E)$ be a graph consisting of two components isomorphic to P_3. For each $u \in V$, let $\varphi(u) = \nu$ and $\pi_i(u) = p$ if $deg(u) = 1$, otherwise $\pi_i(u) = q$ if $deg(u) = 2$. Let $\pi_s(e) = 1$ for each $e \in E$. Let H_1 be a cut system consisting of two edges from the same component and let H_2 be a cut system of two edges one from each component. Then $\rho(G_{H_2}) < \rho(G_{H_1})$ if $0 < p < \frac{2}{3}$, $0 \leq q < 1$, and $\rho(G_{H_1}) - \rho(G_{H_2}) = p(2 - 3p)(1 - q)\nu$.*

Proof. The initial graph G and the two graphs resulting from the application of the cut systems H_1 and H_2 are depicted in Figure A.1. Let us calculate the risks $\rho(G_{H_1})$ and $\rho(G_{H_2})$, as a direct application of Proposition 1.1.3:

$$\rho(G_{H_1}) = \nu(2p + q) + 3\nu(1 - (1 - p)^2(1 - q))$$
$$\rho(G_{H_2}) = 2\nu(p + 2(1 - (1 - p)(1 - q))).$$

By dividing the risks by the node value ν and then expanding the equations, we obtain:

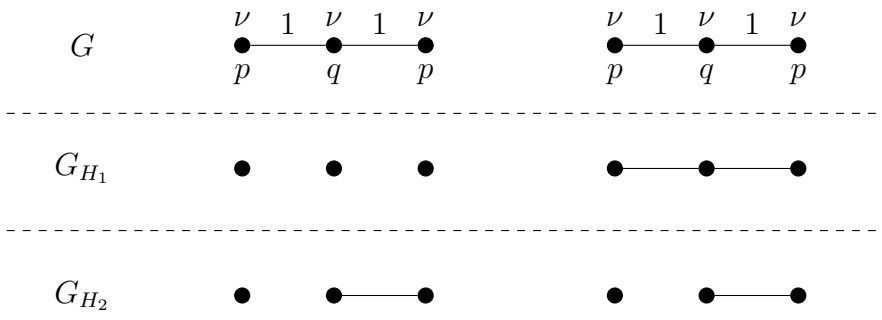


Figure A.1: Graphs G and the two graphs G_{H_1} and G_{H_2}, resulting from the application of the cut systems H_1 and H_2, respectively. Vertices in G (up) are labeled with their probabilities of ignition (below) p and q and their value ν (above). Edges in G are labeled with their probability of spread.

$$\frac{\rho(G_{H_1})}{\nu} = 2p + q + 3(2p + q - p^2 - 2pq + p^2q) = 8p + 4q - 3p^2 - 6pq + 3p^2q$$
$$\frac{\rho(G_{H_2})}{\nu} = 2(p + 2(p + q - pq)) = 6p + 4q - 4pq.$$

The difference between the two risks, normalized by ν, is then:

$$\frac{\rho(G_{H_1})}{\nu} - \frac{\rho(G_{H_2})}{\nu} = 2p - 3p^2 - 2pq + 3p^2q = p(2 - 3p)(1 - q).$$

Such a difference is strictly positive and therefore $\rho(G_{H_2}) < \rho(G_{H_1})$ when:

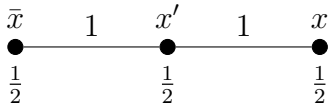
$$0 < p < \frac{2}{3}$$
$$0 \leq q < 1.$$

□

From now, we assume that $0 < p < \frac{2}{3}$ and $0 \leq q < 1$ to ensure we can apply Lemma A.2.3.

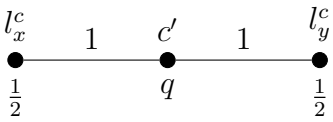
A.2.3 The reduction

In this paragraph, we describe our polynomial reduction from Restricted Strong Planar Max 2SAT to Windy Firebreak Location. In the Figures A.2, A.3,



(a) Variable Path

Vertices have value ν and edges cost s



(b) Clause Path

Vertices have value ω and edges cost 1

Figure A.2: The path P_3 representing (a) a variable x and (b) a clause $c = (l_x^c, l_y^c)$ in an instance of Windy Firebreak Location. The numbers above the edges correspond to the probabilities of spread. Vertex labels are indicated above the vertices while their probabilities of ignition are mentioned below.

A.4, A.5, and A.6, vertices are labeled with their name (unless unnecessary for a good understanding) and the related probability of ignition, when this information is useful. Edges are labeled with the related probability of spread, when useful. To simplify the figures, the values of vertices and the costs of edges are not reported but directly indicated in the text, when relevant. Finally, for a variable z, $l_z \in \{z, \bar{z}\}$ denotes a literal on z.

Let $I = (\Phi, K)$, with $\Phi = (X, C)$, be an instance of Restricted Strong Planar Max 2SAT with n variables and m 2-clauses. We build an instance $(\Gamma^I, \mathbb{1}, \pi_i, \kappa, \varphi, B, R)$ of Windy Firebreak Location, starting from (Φ, K) and the related graph $\widetilde{G}_\Phi$, as follows.

Variables In Γ^I, each variable x is represented in $\widetilde{G}_\Phi$ by a path P_3 $xx'\bar{x}$ (see Figure A.2-a). For each vertex u of these P_3 paths, we set $\pi_i(u) = \frac{1}{2}$ and $\varphi(u) = \nu$. For each edge e of these P_3 paths, we set $\pi_s(e) = 1$ and $\kappa(e) = s$. As in $\widetilde{G}_\Phi$, for each variable $x \in X$, the external vertices of the corresponding P_3 represent the literals x and $\bar{x}$. We refer to these paths as *variable paths*.

Clauses As for the clauses, for each vertex c of $\widetilde{G}_\Phi$ representing a clause $c = (l_x^c, l_y^c) \in C$, there is a path P_3 denoted by $l_x^c c' l_y^c$ in Γ^I whose external vertices represent the literals l_x^c and l_y^c (see Figure A.2-b). For each vertex u of these P_3 paths we define $\varphi(u) = \omega$, whereas $\pi_i(u) = \frac{1}{2}$ if u is an external vertex of the corresponding

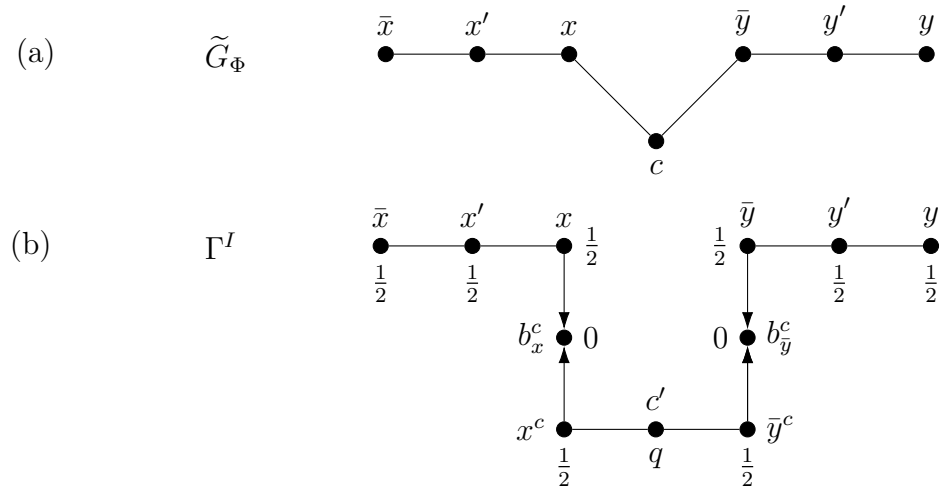


Figure A.3: (a) represents the clause $(x, \bar{y})$ in $\widetilde{G}_\Phi$. (b) represents how the two related variable paths and the related clause path are connected in Γ^I. Probabilities of ignition are indicated on vertices in Γ^I. Probabilities of spread are all 1, $\varphi(b_x^c) = \varphi(b_{\bar{y}}^c) = 1$ and $\kappa(xb_x^c) = \kappa(x^c b_x^c) = \kappa(\bar{y}b_{\bar{y}}^c) = \kappa(\bar{y}^c b_{\bar{y}}^c) = B+1$. All other vertex values and edge costs in Γ^I are as in Figure A.2.

P_3, $\pi_i(u) = q$ otherwise. For each edge e of these P_3s, we set $\pi_s(e) = 1$ and $\kappa(e) = 1$. We denote these paths as *clause paths*.

Connection between literals and clauses Variables and clauses are then connected as follows. For each edge cl_x in $\widetilde{G}_\Phi$, where $l_x \in \{x, \bar{x}\}$ is a literal appearing in c, we introduce a *binding* vertex $b_{l_x}^c$ in Γ^I such that $\pi_i(b_{l_x}^c) = 0$ and $\varphi(b_{l_x}^c) = 1$. Then, for each binding vertex $b_{l_x}^c$, with $l_x \in \{x, \bar{x}\}$, there are two directed edges $e_1 = l_x b_{l_x}^c$ and $e_2 = l_x^c b_{l_x}^c$ in Γ^I such that $\pi_s(e_1) = \pi_s(e_2) = 1$ and $\kappa(e_1) = \kappa(e_2) = B+1$, as shown in Figure A.2-b. Figure A.3 gives an example of graphs $\widetilde{G}_\Phi$ and Γ^I when C includes only the clause $c = (x, \bar{y})$. Note that all the probabilities of spread are set to 1 and then the built instance is a Windy Firebreak Location instance. Note that the related graph is planar and bipartite of maximum degree 5.

The resulting graph has $(3n + 5m)$ vertices and $(2n + 6m)$ edges.

To conclude the construction, we set $B = ns + m$ and $R = 2n\nu + m\omega\left(\frac{3}{2} + q\right) + m\left(\frac{7}{4} + \frac{q}{8}\right) - \frac{K}{8}$. Note that, since $K \leq m$, we have $R > 0$.

A.2.4 Choice of parameters and related properties

Our objective is to show that, for certain values of the parameters s, ν, ω and q, I is satisfiable if and only if there is a cut system H for Γ^I such that $\kappa(H) \leq B$ and $\rho(\Gamma_H^I) \leq R$. In this paragraph, we determine the parameters and establish their

main properties.

Choice of s First, we choose $s = m + 1$; this ensures:

Claim A.2.4. *Any cut system H for Γ^I such that $\kappa(H) \leq B$ has at most n edges from the variable paths.*

Proof. This is due to the choice of B: if H has $n + 1$ cuts on the variable paths, then $\kappa(H) \geq (n + 1)s = ns + s > ns + m = B$. □

Choice of ν We set $\nu = 8m\left(\frac{5}{2}\omega + 2\right)$.

Then, the choices of s and ν guarantee the following claim:

Claim A.2.5. *Any cut system H for Γ^I such that $\kappa(H) \leq B$ and $\rho(\Gamma^I_H) \leq R$ uses n cuts on the edges of the variable paths, one for each variable. The related induced cost is ns and the induced risk is $2n\nu$.*

Proof. We assume that n_i variable paths have i cuts for $i = 0, 1, 2$: $n_0 + n_1 + n_2 = n$ and the related number of cuts is $n_1 + 2n_2$. Note first that, if at most n cuts are performed on variable paths, then $n_2 \leq n_0$ and then, Lemma A.2.3 ensures that, while $n_2 > 0$, pairing one variable path with two cuts with one with no cut and transferring one cut from the former to the latter allows to reduce the risk without changing the number of cuts used on the n variable paths.

Assume first that no more than $n - 1$ cuts are performed on variable paths. Then, $n_0 \geq n_2 + 1$ and the previous remark ensures that the smallest contribution to the total risk induced by vertices from variable paths is obtained for $n_2 = 0$, with still $n_0 \geq 1$. Then, Proposition 1.1.3 ensures that the related contribution to the risk is $(n - n_0)\nu\left(\frac{1}{2} + 2\left(1 - (\frac{1}{2})^2\right)\right) = 2\nu(n - n_0)$ for the $(n - n_0)$ variable paths with one cut and a contribution of $3n_0\nu(1 - (\frac{1}{2})^3) = \frac{21}{8}n_0\nu$ for the n_0 remaining paths with no cut. So, since $n_0 \geq 1$, the contribution to the total risk induced by the variable paths is at least $(2(n - 1) + \frac{21}{8}) \cdot \nu$.

$$\begin{aligned}
\left(2(n-1) + \frac{21}{8}\right) \cdot \nu &= 2n\nu + 10m + \frac{25}{2}m\omega \\
&\geq 2n\nu + \frac{5}{2}m\omega + 2m \\
&\geq 2n\nu + m\omega\left(\frac{3}{2} + 1\right) + 2m \\
&> 2n\nu + m\omega\left(\frac{3}{2} + q\right) + m\left(\frac{7}{4} + \frac{q}{8}\right) - \frac{K}{8} \\
&> R.
\end{aligned}$$

where the strict inequality comes from $q < 1$.

So at least n cuts are performed on variable paths to ensure a risk no more than R. Claim A.2.4 induces that exactly n cuts are performed on variable paths.

We now need to guarantee that the n cuts are distributed exactly as one for each variable path. We now have $n_0 = n_2$ and Lemma A.2.3 shows that the risk induced by the variable paths decreases with n_0. If $n_0 = 1$, then Proposition 1.1.3 shows that this risk is $2\nu n + \frac{1}{2}\left(2 - 3\frac{1}{2}\right)\left(1 - \frac{1}{2}\right)\nu = 2n\nu + \frac{\nu}{8}$.

In this case, using a similar argument as before, the risk is at least:

$$\begin{aligned} 2n\nu + \frac{\nu}{8} &= 2n\nu + m\left(\frac{5}{2}\omega + 2\right) \\ &= 2n\nu + m\omega\left(\frac{3}{2} + 1\right) + 2m \\ &> 2n\nu + m\omega\left(\frac{3}{2} + q\right) + m\left(\frac{7}{4} + \frac{q}{8}\right) - \frac{K}{8} \\ &> R. \end{aligned}$$

This implies that we have $n_0 = n_2 = 0$ to ensure a risk at most R. In this case, Proposition 1.1.3 ensures that the risk induced on vertices of variable paths is $2n\nu$. This concludes the proof. □

Choice of ω We set $\omega = \frac{8m}{1-q}$; in addition to the previous choices of s and ν, it ensures:

Claim A.2.6. *Any cut system H for Γ^I such that $\kappa(H) \leq B$ and $\rho(\Gamma^I_H) \leq R$ uses one cut per clause path.*

Proof. Claim A.2.5 ensures that the cost of cuts on variables paths is ns and the related risk is $2n\nu$. So, the budget m remains available and can be used for m cuts on the edges of the clause paths in such a way that the related contribution to the risk does not exceed $R - 2n\nu$.

As before, we denote by m_i the number of clause paths with i cuts, $i = 0, 1, 2$. We have $m_2 = m_0$ to ensure m cuts on these paths.

Assume that $m_2 > 0$; Lemma A.2.3 guarantees that we can reduce the risk by replacing one clause path with two cuts and one with no cut by two clause paths with one cut each. Thus, the minimum risk induced by vertices in clause paths with $m_2 > 0$ is obtained for $m_2 = m_0 = 1$.

Proposition 1.1.3 ensures that the risk induced when all m clause paths have a single cut is $m\omega(\frac{1}{2} + 2(\frac{1}{2} + q - \frac{1}{2}q)) = m\omega(\frac{3}{2} + q)$. Then, using Lemma A.2.3, if two clause paths with one cut are replaced by one with two cuts and one without any cut, then the risk increase is $\frac{1}{2}\left(2 - 3\frac{1}{2}\right)(1 - q)\omega = \frac{\omega}{4}(1 - q)$.

Our choice of ω ensures $\frac{\omega}{4}(1-q) \geq 2m$ and then the minimum risk induced if $m_2 > 0$ is:

$$m\omega\left(\frac{3}{2}+q\right)+\frac{\omega}{4}(1-q) \geq m\omega\left(\frac{3}{2}+q\right)+2m > R-2n\nu \tag{A.2}$$

So, we need $m_2 = 0$ to ensure that clause paths have a contribution of the risk at most $R - 2n\nu$, which concludes the proof. □

Choice of q We finally choose: $q = 1 - \frac{1}{2K-1}$. This implies:

$$q = 2 - \frac{K}{K-\frac{1}{2}} \Leftrightarrow \frac{K}{2-q} = K - \frac{1}{2} \tag{A.3}$$

that will be used to ensure the validity of the reduction.

A.2.5 Validity of the reduction

In this paragraph, we justify that our reduction is valid.

Claim A.2.7. *The instance of* Restricted Strong Planar Max 2SAT*, $I = (\Phi, K)$, with $\Phi = (X, C)$ is satisfiable if and only if the instance $(\Gamma^I, \mathbb{1}, \pi_i, \kappa, \varphi, B, R)$ of* Windy Firebreak Location *admits a cut system H for Γ^I such that $\kappa(H) \leq B$ and $\rho(\Gamma^I_H) \leq R$.*

Proof. Assume that the instance I of Restricted Strong Planar Max 2SAT is satisfiable, i.e., the instance I given by (X, C, K), has an assignment for the variables in X such that at least K clauses are satisfied. Then, we prove that there exists a cut system H for Γ^I such that $\kappa(H) \leq B$ and $\rho(\Gamma^I_H) \leq R$.

The sought cut system H cuts each variable path on the side of the true literal. With this assumption, we evaluate now the induced risk on the binding vertices.

We denote with K_i the number of clauses with i satisfied literals, $i = 0, 1, 2$. Then $K_1 + K_2 \geq K$ and $K_0 + K_1 + K_2 = m$.

Let ρ_i be the induced risk on the binding vertices, for all clauses with i true literals. Then the total induced risk on the binding vertices is $\rho = \rho_0 + \rho_1 + \rho_2$. The subgraphs of Γ^I_H representing a clause $c = (l_x, l_y)$ for all the possible assignments to l_x and l_y are shown in Figures A.4, A.5, and A.6. Note that a cut on a clause path can be on any of the two edges. When l_x and l_y have the same assignment, the subgraphs resulting by cutting the clause path either on the left-hand side or on the right-side are isomorphic (see Figures A.4 and A.6), whereas, if l_x is False and l_y is True, i.e., the clause has only one satisfied literal, then the two possible subgraphs, depending on which edge of the clause path is cut, are not isomorphic (see Figure A.5).

When l_x and l_y are both False (see Figure A.4), the risk for the binding vertex on the left is $1-(1-1/2)(1-1/2)(1-1/2) = 7/8$, whereas, for the vertex on the right

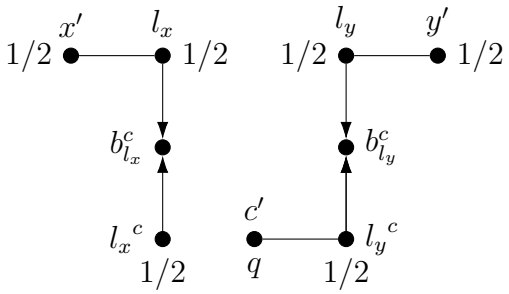
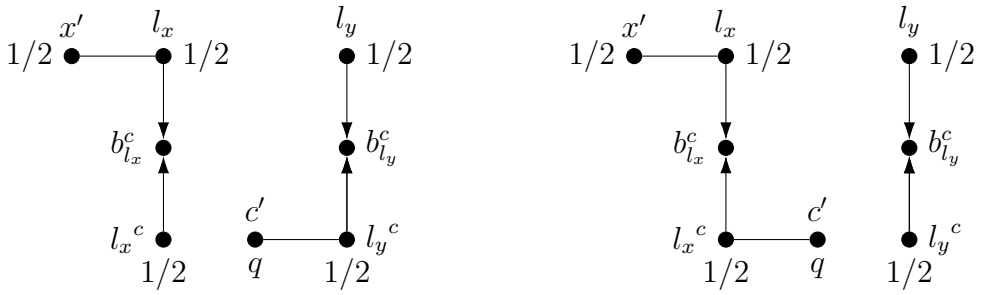


Figure A.4: Clause $c = (l_x, l_y)$ with l_x = False and l_y = False. Example where the clause path is cut on the left-hand side; if it is cut on the right-hand side, then the resulting partial graph is isomorphic.



(a) Clause Path cut on the left-hand side

(b) Clause Path cut on the right-hand side

Figure A.5: Clause $c = (l_x, l_y)$ with l_x = False and l_y = True and the two possible cuts of the clause path. This time, the resulting partial graphs are not isomorphic.

it is $1 - (1 - 1/2)(1 - 1/2)(1 - 1/2)(1 - q) = 1 - 1/8 + q/8 = 7/8 + q/8$. In total, we have $7/4 + q/8$. Then:

$$\rho_0 = (7/4 + q/8) \cdot K_0. \tag{A.4}$$

When l_x and l_y are both True (see Figure A.6), the risk for the binding vertex on the left is $1 - (1 - 1/2)(1 - 1/2) = 3/4$, whereas, for the vertex on the right it is $1-(1-1/2)(1-1/2)(1-q) = 1-1/4+q/4 = 3/4+q/4$. In total, we have $3/2+q/4$. Then:

$$\rho_2 = (3/2 + q/4) \cdot K_2. \tag{A.5}$$

When l_x is False and l_y is True, we have two cases, as depicted in Figure A.5-a and Figure A.5-b. In the first case (Figure A.5-a): the risk for the binding vertex on the left is $1 - (1 - 1/2)(1 - 1/2)(1 - 1/2) = 7/8$, whereas, for the vertex on the right it is $1 - (1 - 1/2)(1 - 1/2)(1 - q) = 1 - 1/4 + q/4 = 3/4 + q/4$, and in total we have $13/8+q/4$. In the second case (Figure A.5-b): the risk for the binding vertex on the left is $1 - (1 - 1/2)(1 - 1/2)(1 - 1/2)(1 - q) = 1 - 1/8 + q/8 = 7/8 + q/8$, whereas,

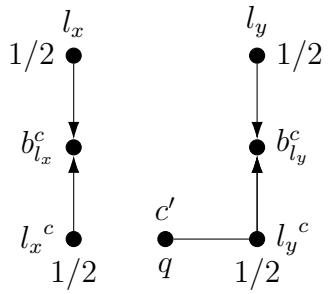


Figure A.6: Clause $c = (l_x, l_y)$ with l_x = True and l_y = True. Example where the clause path is cut on the left-hand side; if it is cut on the right-hand side, then the resulting partial graph is isomorphic.

for the vertex on the right it is $1 - (1 - 1/2)(1 - 1/2) = 3/4$, and in total we have $13/8 + q/8$. Then, by choosing opportunistically the cut that minimizes these two values on the clause path representing the clause (l_x, l_y), we have:

$$\rho_1 = (13/8 + q/8) \cdot K_1. \tag{A.6}$$

Since $K_0 = m - (K_1 + K_2)$, we deduce from Equations A.4, A.5, and A.6:

$$\begin{aligned} \rho = \rho_0 + \rho_1 + \rho_2 \quad &= m(\tfrac{7}{4} + \tfrac{q}{8}) - (K_1 + K_2)(\tfrac{7}{4} + \tfrac{q}{8}) + (\tfrac{13}{8} + \tfrac{q}{8})K_1 + (\tfrac{3}{2} + \tfrac{q}{4})K_2 \\ &= m(\tfrac{7}{4} + \tfrac{q}{8}) - \tfrac{K_1}{8} + (-\tfrac{1}{4} + \tfrac{q}{8})K_2. \end{aligned} \tag{A.7}$$

Since $q \leq 1$, we deduce $\rho \leq m(\frac{7}{4} + \frac{q}{8}) - \frac{(K_1+K_2)}{8} \leq m(\frac{7}{4} + \frac{q}{8}) - \frac{K}{8}$.

Recall that, using Proposition 1.1.3, the total induced risk on variable vertices is $2n\nu$ (see proof of Claim A.2.5) and the one on clause vertices is $m\omega(\frac{3}{2} + q)$ (see proof of Claim A.2.6). Then, the total risk is:

$$\rho(\Gamma^I_H) = 2n\nu + m\omega\left(\frac{3}{2} + q\right) + \rho \leq 2n\nu + m\omega\left(\frac{3}{2} + q\right) + m\left(\frac{7}{4} + \frac{q}{8}\right) - \frac{K}{8} = R.$$

Conversely, let us assume that the instance $(\Gamma^I, \mathbb{1}, \pi_i, \kappa, \varphi, B, R)$ of Windy Firebreak Location admits a cut system H for Γ^I such that $\kappa(H) \leq B$ and $\rho(\Gamma^I_H) \leq R$. Then we prove that the instance $I = (X, C, K)$ of Restricted Strong Planar Max 2SAT, has an assignment for the variables in X such that at least K clauses are satisfied.

Claim A.2.5 ensures that H has necessarily one cut in each variable path and Claim A.2.6 ensures that it has one cut in each clause path. The induced risk of the vertices in variable paths is $2n\nu$, while the induced risk of the vertices in

clause paths is $m\omega(\frac{3}{2}+q)$. Since the total risk is less than R, the risk ρ' induced on the binding vertices satisfies

$$\rho' \leq m\left(\frac{7}{4}+\frac{q}{8}\right)-\frac{K}{8}. \tag{A.8}$$

By using the same notation K_0, K_1, K_2 as above and since Equation A.7 gives the minimum possible risk for fixed K_0, K_1, K_2,we deduce

$$\begin{array}{rl} \rho' & \geq m(\frac{7}{4}+\frac{q}{8})-\frac{K_1}{8}+(-\frac{2}{8}+\frac{q}{8})K_2 \\ & \geq m(\frac{7}{4}+\frac{q}{8})-\frac{K1+(2-q)K2}{8} \\ & \geq m(\frac{7}{4}+\frac{q}{8})-(2-q)\frac{K1+K2}{8}. \end{array} \tag{A.9}$$

We deduce from Inequalities A.8 and A.9 that $m(\frac{7}{4}+\frac{q}{8})-\frac{K1+K2}{8}(2-q) \leq m(\frac{7}{4}+\frac{q}{8})-\frac{K}{8}$. Then, Equation A.3 implies:

$$K_1+K_2 \geq \frac{K}{2-q}. \tag{A.10}$$

This implies $K_1+K_2 \geq K-\frac{1}{2}$. Since K, K_1, K_2 are integers, this implies $K_1+K_2 \geq K$ and thus, I is a positive instance. This concludes the proof.

□

A.2.6 The main result

We are now ready to prove the main result of this section.

Proposition A.2.8. Restricted Strong Planar Max 2SAT *polynomially reduces to* Windy Firebreak Location *in bipartite planar graphs of maximum degree 5 and with polynomially bounded vertex values and edge costs.*

Proof. We use the reduction from Restricted Strong Planar Max 2SAT described in Paragraph A.2.3 with parameters defined in Paragraph A.2.4. From an instance of Restricted Strong Planar Max 2SAT with n variables and m clauses, it constructs an instance of Windy Firebreak Location with $(3n+5m)$ vertices and $(2n+6m)$ edges. The related graph is planar, bipartite and of maximum degree 5. Since parameters can all be computed in constant time, the construction is polynomial. Note finally that the edge costs in the reduction are 1 or s and the vertex values are $1, \nu$ or ω. All these values are integral and polynomially bounded with respect to n and m and thus, to the size of the Windy Firebreak Location instance. Indeed, $s = m+1$, $\omega = \frac{8m}{1-q}$ and with $q = 1 - \frac{1}{2K-1}$, we get $\omega = 8m(2K-1)$ with $K \leq m$. Finally, $\nu = 8m\left(\frac{5}{2}\omega + 2\right) = 20m\omega + 16m$.

Claim A.2.7 justifies this is a valid reduction, which concludes the proof. □

Since Windy Firebreak Location is NP-complete (see Corollary A.1.10), we immediately deduce:

Corollary A.2.9. Windy Firebreak Location *is NP-complete on bipartite planar graphs of maximum degree 5 and with polynomially bounded vertex values and edge costs.*

A.3 Some refinements in the instances

The proof of Proposition A.2.8 is written using different values on vertices to keep it as simple as possible. In what follows, we use self-refinements to show that the case with integral and polynomially bounded vertex values and edge costs polynomially reduces to the case where all vertices have the same value 1 and all edges have the same cost 1.

In Paragraph A.3.1, we first show how to reduce the vertex values to 1. Then, in Paragraph A.3.2, we describe how to reduce the edge costs to 1. Finally, in Paragraph A.3.3 we combine both techniques to establish our main result.

A.3.1 Reducing vertex values

Proposition A.3.1. *For any graph class $\mathcal{C}$ closed under subdivision of edges,* Windy Firebreak Location *with polynomially bounded vertex values, polynomially re-*

duces to its particular case where all vertices have value 1. The transformation preserves the maximum degree.

Proof. Consider an instance $I = (\Gamma, \mathbb{1}, \pi_i, \kappa, \varphi, B, R)$ of Windy Firebreak Location, where $\Gamma = (V, E) \in \mathcal{C}$ and we assume that there is a polynomial P such that $\forall v \in V, \varphi(v) \leq P(|V|)$. We build in polynomial time an instance $I' = (\Gamma', \mathbb{1}, \pi'_i, \kappa', \varphi_{\mathbb{1}}, B, R)$ with $\Gamma' = (V', E') \in \mathcal{C}$, $\varphi_{\mathbb{1}}$ is the constant function on V' that maps any vertex to 1 and I' is positive if and only if I is positive. The budget and the risk threshold remain unchanged.

For every vertex $v \in V$ of degree d, We denote by $u_1, \ldots, u_d$ the neighbors of v and insert $\mu_i \geq 0$ vertices on the edge vu_i, $i = 1, \ldots, d$ such that $1 + \sum_{i=1,\ldots,d} \mu_i = \varphi(v)$. We denote by $X(v)$ the set of new vertices; all edges between two vertices in $\{v\} \cup X(v)$ are non-directed and have the same cost $B + 1$. Vertices in $X(v)$ have an probability of ignition 0 while the probability of ignition of v is unchanged: $\pi'_i(v) = \pi_i(v)$. The edge vu_i is replaced by the edge $z_i^v u_i$, where z_i^v is the vertex inserted on vu_i that is linked to u_i (could be v if $\mu_i = 0$). If vu_i is directed, then $z_i^v u_i$ is directed with the same orientation. If we perform this transformation for every vertex, then we get an instance I' in a graph $\Gamma' = (V', E') \in \mathcal{C}$, where all vertices have the same value 1. We can see V and E as subsets of V' and E', respectively and there is a one to one correspondence between edges of cost at most B in Γ and in $\Gamma' = (V', E')$ and the cost is preserved by this correspondence. In particular, any cut system $H \subset E$ in Γ such that $\kappa(H) \leq B$ can be seen as a cut system in Γ' with the same cost. It is straightforward that $\rho(\Gamma_H) = \rho(\Gamma'_H)$. Indeed, if a vertex v burns with some probability in Γ_H, then, in Γ'_H the $\varphi(v)$ vertices in $\{v\} \cup X(v)$ will burn with the same probability. Since the transformation can be performed in polynomial time, the proof is complete. □

The transformation in Proposition A.3.1 preserves planarity and the maximum degree. However, it does not necessarily preserve bipartite graphs. Note however that bipartiteness can easily be imposed. Multiplying all vertex values by the same number just induces multiplying the risk of any solution by this constant; so, it does not change the problem. Then, we propose a first transformation ensuring that all vertices have an odd degree. First multiply all vertex values as well as the risk threshold by 2 to ensure all values are at least 2. If a vertex v has an even degree, then add a pending vertex of value 1 and probability of ignition 0, reduce the value of v by 1 and define the cost of the related edge as $B+1$. This defines an equivalent instance where all vertices have an odd degree. In addition if the maximum degree is odd, this operation does not change it, otherwise it adds 1 to the maximum degree. Denote now by Δ the maximum degree in the new graph and multiply all vertex values as well as the risk threshold by 2Δ to obtain an equivalent instance where each vertex has an even value greater than its degree. In this case, we can always perform the transformation in Proposition A.3.1 ensuring that all μ_is are odd. For instance, choose all μ_is but one equal to 1. This ensures the new graph Γ' is bi-

partite. Note finally that all edges in Γ are preserved with their cost and the new edges have the cost $B+1$. Without loss of generality we can assume $B \leq \sum_{e \in E} \kappa(e)$ and consequently, if all edge costs are polynomially bounded in Γ, so are they in Γ'. Using this argument we deduce the following corollary:

Corollary A.3.2. WINDY FIREBREAK LOCATION *remains NP-complete in bipartite planar graphs of maximum degree 5 when all the vertices' values are 1 and edge costs are polynomially bounded.*

A.3.2 Reducing edge values

A natural question is whether we can add the constraint that edge costs are all 1. A first answer is that replacing an edge of cost c by c parallel edges, each of cost 1, makes the problem equivalent. The transformation preserves planarity and bipartiteness but it does not preserve low degree. In what follows, we sketch a polynomial reduction that allows to maintain the degree bounded.

Proposition A.3.3. WINDY FIREBREAK LOCATION *with polynomially bounded edge costs, all vertex values 1 and a rational probability system with a polynomial least common multiple, polynomially reduces to* WINDY FIREBREAK LOCATION *with all edge costs and vertex values equal to 1 in a graph of maximum degree 4.*

Proof. Consider an instance $I = (\Gamma, \mathbb{1}, \pi_i, \kappa, \mathbb{1}, B, R)$ of WINDY FIREBREAK LOCATION with all vertex values equal to 1. We assume $\Gamma = (V, E)$ with $n = |V|$, edge costs are integers and probabilities are all rational. We also assume there is a polynomial integral function f such that $\forall e \in E, \kappa(e) \leq f(n)$ and $\forall x \in V, f(n)\pi_i(x) \in \mathbb{N}$. This last property ensures that for all cut system H in Γ, $f(n)\rho(\Gamma_H) \in \mathbb{N}$. To simplify further expressions, we define $C = \left\lceil \frac{B}{2} \right\rceil$. The construction depends on a polynomially bounded value M, chosen as follows:

$$M = \max\left(1 + C\left\lceil\sqrt{2Rf(n)+1}\right\rceil, |E|f(n)\right). \tag{A.11}$$

We build an instance $I' = (\Gamma', \mathbb{1}, \pi'_i, \mathbb{1}, \mathbb{1}, B', R')$ with all edge cost equal to 1 such that I' is positive if and only if I is positive. In addition Γ' has maximum degree 4. Γ' is obtained from Γ by replacing each vertex x with a $(M \times M)$ non-directed square grid Q_x with M^2 vertices. Edges of $Q_x, x \in V$ are called *Q-edges* in Γ'. Each edge (x, y) of cost $\kappa((x, y))$ is replaced with a set $J_{(x,y)}$ of $\kappa((x, y))$ edges, each of cost 1; such edges are called *joining edges* in Γ'. All edges incident to x in Γ correspond, in Γ', to $\sum_{(x,y)\in E} \kappa((x, y))$ joining edges incident to the perimeter of Q_x and with extremities equally spread along this perimeter. Since $M \geq |E|f(n) \geq \sum_{e \in E} \kappa(e)$, the perimeter of Q_x is long enough. All vertices in Q_x have the same probability of ignition equal to $1 - (1 - \pi_x)^{\frac{1}{M^2}}$, where $0^{\frac{1}{M^2}} = 0$. Finally, we define $B' = B$ and

$R' = M^2R$. Note that the maximum degree of Γ' is 4 and that the construction can be performed in polynomial time.

Assume first that I has a cut system $H \subset E$ such that $\kappa(H) \leq B$ and $\rho(G_H) \leq R$. We then define a cut system $H' = \bigcup_{(x,y)\in H} J_{(x,y)}$ in Γ'. With the edge costs in Γ and Γ' we have $\kappa'(H') = \kappa(H) \leq B$. It is straightforward to verify that $\rho(\Gamma'_{H'}) = M^2\rho(\Gamma_H)$ since Q_x's probability of burning in $\Gamma'_{H'}$ is exactly the probability that x burns in Γ_H and $\varphi'(Q_x) = M^2\varphi(x)$. So, I' is positive.

Conversely, assume I' is positive and let H' be a cut system satisfying $\kappa'(H') \leq B$ and $\rho(\Gamma'_{H'}) \leq M^2R$.

For a vertex $x \in V$, a *connected component* of $\Gamma'_{H'}[Q_x]$ is called *small* if its size is at most C^2 and it is *large* instead. Equation A.11 implies in particular $M > B$.

We establish a few claims that, all together, allow to complete the proof of Proposition A.3.3:

Claim A.3.4. $\forall x \in V$, $\Gamma'_{H'}[Q_x]$ *has exactly one large component.*

Proof. Since all edge costs in I' are 1, we have $|H'| \leq B$. Since $M > B$, removing B edges from the $M \times M$ grid Q_x allows to disconnect at most C^2 vertices from the rest of the grid (this maximum is obtained if removed edges disconnect a corner $C \times C$ of the grid Q_x). Equation A.11 implies $M^2 - C^2 > C^2$ and consequently the remaining vertices in Q_x constitute a large component. This concludes the proof of the claim. □

The same argument allows to show:

Claim A.3.5. *The total size of all small components in* $\Gamma'_{H'}$ *is at most* C^2.

To derive from H' a cut system in Γ, we first transform H' into H'' that only includes joining edges:

- Any joining edge in H' is added to H'';
- Any joining edge adjacent to a small component of $\Gamma'_{H'}[Q_x]$ for some $x \in V$ is added to H'' if not yet in it;

Note that a similar argument as in the previous claim shows that the number of Q-edges in H' is at least equal to the number of joining edges adjacent to a small component of $\Gamma'_{H'}[Q_x]$. As a consequence, $|H''| \leq |H'|$.

We denote by V_Q the set of vertices of small components in $\Gamma'_{H'}$ and consider the graph $\Gamma'' = \Gamma'[V \setminus V_Q]$.

Claim A.3.6. $\rho(\Gamma''_{H''}) \leq \rho(\Gamma'_{H'})$.

Proof. Connected components of $\Gamma''_{H''}$ are contained in connected components of $\Gamma'_{H'}$. □

We now define a cut system H in Γ as follows: for every edge $e \in E$, add it in H if and only if $J_e \subset H''$. It is straightforward to show that $\kappa(H) \leq |H''|$.

Claim A.3.7. $\rho(\Gamma_H) \leq \frac{\rho(\Gamma''_{H''})}{M^2-2C^2}$.

Proof. Since all vertices have value 1, we deduce from Proposition 1.1.3 that $\rho(\Gamma_H) = \sum_{x \in G_H} p_x$. Consider a vertex $x \in V$ and the related vertex set $Q_x \setminus V_Q$ in Γ''. All vertices in $Q_x \setminus V_Q$ are connected in $\Gamma''_{H''}$ due to Claim A.3.4.

Consider an edge (y, x) in Γ_H. By definition of H, there is at least one edge from $Q_y \setminus V_Q$ to $Q_x \setminus V_Q$. So, let $z_x \in Q_x \setminus V_Q$, $y \in U_{x,H}$ in Γ and $z_y \in Q_y \setminus V_Q$. We have $z_y \in U_{z_x,H''}$ in Γ''. Conversely, if $z_y \in U_{z_x,H''}$ in Γ'' with $z_x \in Q_x \setminus V_Q$ and $z_y \in Q_y \setminus V_Q$, then $y \in U_{x,H}$ in Γ.

If we call p''_{z_x} the probability that $z_x \in Q_x \setminus V_Q$ burns in $\Gamma''_{H''}$, we have:

$$\begin{aligned} 1 - p''_{z_x} &= \prod_{t \in U_{z_x,H''}} (1 - \pi_i(t)) \\ &\leq \prod_{y \in U_{x,H}} (1 - \pi_i(y))^{\frac{M^2-C^2}{M^2}} \\ &= (1 - p_x)^{\frac{M^2-C^2}{M^2}}. \end{aligned}$$

If $p_x < 1$, we consider the real function $h : z \mapsto (1 - p_x)^z = e^{z \log(1-p_x)}$. It is decreasing and convex. As a consequence, on the interval $[0, 1]$, h is bounded above by the linear function equal to $h(0) = 1$ for $z = 0$ and equal to $h(1) = (1 - p_x)$ for $z = 1$. So, $\forall 0 \leq z \leq 1$, $(1-p_x)^z \leq 1 - zp_x$. If $p_x = 1$, then the inequality also holds.

Since, for any vertex $x \in V$ in Γ, the large component associated with x in Γ'' includes at least $M^2 - C^2$ vertices, we have:

$$\begin{aligned} \rho(\Gamma''_{H''}) &\geq (M^2 - C^2) \sum_{x \in V} \left(1 - (1 - p_x)^{\frac{M^2-C^2}{M^2}}\right) \\ &\geq (M^2 - C^2) \sum_{x \in V} \left((1 - \tfrac{C^2}{M^2})p_x\right) \\ &\geq (M^2 - C^2)(1 - \tfrac{C^2}{M^2})\rho(\Gamma_H) \\ &\geq (M^2 - 2C^2)\rho(\Gamma_H). \end{aligned}$$

We deduce:

$$\rho(\Gamma_H) \leq \frac{\rho(\Gamma''_{H''})}{M^2 - 2C^2}.$$

which completes the proof of the claim. □

Using Claim A.3.6, Claim A.3.7 and $\rho(\Gamma'_{H'}) \leq M^2R$ we deduce:

$$\begin{aligned} \rho(\Gamma_H) &\leq \tfrac{M^2R}{M^2-2C^2} \\ &\leq R + \tfrac{2C^2R}{M^2-2C^2}. \end{aligned} \tag{A.12}$$

Equation A.11 implies $\frac{2C^2R}{M^2-2C^2} < \frac{1}{f(n)}$. So, Inequality A.12 implies that $\rho(\Gamma_H) < R + \frac{1}{f(n)}$. Since $f(n)\rho(\Gamma_H) \in \mathbb{N}$, we deduce $\rho(\Gamma_H) \leq R$, and consequently I is positive, which completes the proof of Proposition A.3.3. □

A.3.3 All together: main result

To complete Section A, we combine techniques used in Paragraphs A.3.1 and A.3.2 to establish Theorem 1.2.2, our main complexity result.

The transformation in Proposition A.3.3 preserves planarity. We can easily ensure that it preserves bipartiteness. Assume indeed that the original graph $\Gamma = (V, E)$ is bipartite with two parts black and white and consider for instance a black vertex $x \in V$. We then consider any black and white partition of the grid Q_x and branch joining edges along the perimeter of Q_x only on black vertices. Doing it for any x ensures that Γ' is bipartite. Since $M \geq \sum_{e \in E} \kappa(e)$, the perimeter is long enough.

In the proof of Proposition A.2.8, the reduction involves a class of Windy Firebreak Location instances with only three different probabilities of ignition, 0, $\frac{1}{2}$ and $q = 1 - \frac{1}{2K-1}$, where K can be chosen not greater than the number of clauses which is less than the number of vertices in the instance of Windy Firebreak Location. Edge costs are either 1, $m+1$ or $(n+1)(m+1)$, where both n and m are less than the number of vertices in the instance of Windy Firebreak Location. So, we can choose $f(n) = 2(2K-1)(n+1)(m+1)$ that satisfies all requirements of Proposition A.3.3. Applying successively the reductions in Proposition A.2.8, Corollary A.3.2 and Proposition A.3.3 from an instance (Φ, K) of Restricted Strong Planar Max 2SAT allows to prove the following Theorem that constitutes the main result of this section:

Theorem 1.2.2. Windy Firebreak Location *is NP-complete in bipartite planar graphs of maximum degree 4 with all vertex values and edge costs equal to 1.*

Note that, in Proposition A.3.3 the probability system π'_i for the Windy Firebreak Location with all edge costs and vertex values equal to 1, in a graph of maximum degree 4, is not necessarily rational and it is not polynomially bounded. We let as an open problem the complexity of Windy Firebreak Location in bipartite planar graphs with all vertex values and edges costs equal to 1 and rational and polynomially bounded probabilities of ignition.

We conclude this section with a remark about the complexity in grid graphs that are natural when considering a homogeneous landscape divided into regular square-like

areas. A planar graph of maximum degree 4 can be embedded in polynomial time into a grid [140] in such a way that vertices map to vertices in the grid and edges map to non-crossing paths in the grid. We can also easily ensure that the embedding is a subgraph of the grid, also called a *subgrid*. In other words, there is a subdivision of the original graph that is a subgrid and the construction can be performed in polynomial time. If the new vertices (produced in the subdivision) are all of value 0 and all edges have the same cost 1, then Windy Firebreak Location in this subgrid is equivalent to Windy Firebreak Location in the original graph. Now, if we consider any grid that contains the subgrid as a subgraph, assign the value 1 and the probability of ignition $\pi_i = 0$ to the added vertices and the cost 0 to the added edges. Then, Windy Firebreak Location in this grid is equivalent to Windy Firebreak Location in the subgrid since we can cut any edge of value 0 without changing the total cost. As a result, we immediately deduce:

Proposition 1.2.3. Windy Firebreak Location *is NP-complete if:*

- *the graph is a subgrid with binary vertex values and unitary edge costs;*
- *the graph is a grid with binary vertex values and edge costs.*

The case with binary probabilities of ignition would be as well an interesting open case.

Bibliography

[9] Polycyclic aromatic hydrocarbon. Available online: `https://en.wikipedia.org/wiki/Polycyclic_aromatic_hydrocarbon`. Accessed: 2021-04-07.

[10] Opendem, 2022, Available online: `https://www.opendem.info/index.html`. Accessed: 2022-5-10.

[11] F. Afghah, A. Razi, J. Chakareski, and J. Ashdown. Wildfire monitoring in remote areas using autonomous unmanned aerial vehicles. In *IEEE INFOCOM 2019*, pages 835–840, 2019.

[12] V. Agafonkin. Leaflet - a JavaScript library for interactive maps, 2019, Available online: `https://leafletjs.com/`. Accessed: 2022-5-10.

[13] C. Agathangelou, C. Georgiou, and M. Mavronicolas. A distributed algorithm for gathering many fat mobile robots in the plane. In *Proc. 32nd ACM Symp. on Principles of Distributed Computing (PODC)*, 2013.

[14] H. Ahmadzadeh, E. Masehian, and M. Asadpour. Modular robotic systems: Characteristics and applications. *J. Intell. Robotic Syst.*, 81(3-4):317–357, 2016.

[15] A. Aljohani, P. Poudel, and G. Sharma. Complete visitability for autonomous robots on graphs. In *2018 IEEE International Parallel and Distributed Processing Symposium, IPDPS 2018*, pages 733–742. IEEE Computer Society, 2018.

[16] I. L. H. Alsammak, M. A. Mahmoud, H. Aris, M. AlKilabi, and M. N. Mahdi. The use of swarms of unmanned aerial vehicles in mitigating area coverage challenges of forest-fire-extinguishing activities: A systematic literature review. *Forests*, 13(5), 2022.

[17] M. Amir and A. M. Bruckstein. Minimizing travel in the uniform dispersal problem for robotic sensors. In *Proc. 18th Int.'l Conf. on Autonomous Agents and MultiAgent Systems (AAMAS)*, pages 113–121. International Foundation for Autonomous Agents and Multiagent Systems, 2019.

[18] V. Auletta, G. D. Nittis, D. Ferraioli, N. Gatti, and D. Longo. Strategic monitor placement against malicious flows. In *ECAI 2020 - 24th European Conference on Artificial Intelligence, 2020*, volume 325 of *Frontiers in Artificial Intelligence and Applications*, pages 11–18. IOS Press.

[19] B. Aydin, E. Selvi, J. Tao, and M. J. Starek. Use of fire-extinguishing balls for a conceptual system of drone-assisted wildfire fighting. *Drones*, 3(1), 2019.

[20] M. Beighley and A. C. Hyde. Systemic Risk and Portugal's forest fire defense strategy. In *Portucel Conf*, 2009.

[21] M. Bennett, S. A. Fitzgerald, B. Parker, M. Main, A. Perleberg, C. C. Schnepf, and R. Mahoney. *Reducing Fire Risk on Your Forest Property | OSU Extension Catalog | Oregon State University.* Pacific Northwest Extension Publication, 2018.

[22] S. Bhagat, S. G. Chaudhuri, and K. Mukhopadhyaya. Formation of general position by asynchronous mobile robots under one-axis agreement. In *Proc. 10th Int.'l WS on Algorithms and Computation (WALCOM)*, volume 9627 of *LNCS*, pages 80–91. Springer, 2016.

[23] C. Blasi, G. Bovio, P. M. Corona, M. Marchetti, and A. Maturani. *Incendi e complessità ecosistemica. Dalla pianificazione forestale al recupero ambientale.* Ministero dell'ambiente e della tutela del territorio. SBI, 2004.

[24] K. Bose, R. Adhikary, M. K. Kundu, and B. Sau. Arbitrary pattern formation on infinite grid by asynchronous oblivious robots. In *Proc. 13th Int.'l Conf. on Algorithms and Computation (WALCOM)*, volume 11355 of *LNCS*, pages 354–366. Springer, 2019.

[25] K. Bose, R. Adhikary, M. K. Kundu, and B. Sau. Arbitrary pattern formation by opaque fat robots with lights. In *Proc. 6th Int.'l Conf. on Algorithms and Discrete Applied Mathematics (CALDAM)*, volume 12016 of *LNCS*, pages 347–359. Springer, 2020.

[26] Q. Bramas and S. Tixeuil. Arbitrary pattern formation with four robots. In *Proc. 20th Int.'l Symp. on Stabilization, Safety, and Security of Distributed Systems (SSS)*, volume 11201 of *LNCS*, pages 333–348. Springer, 2018.

[27] V. E. Brimkov and R. P. Barneva. Computational complexity in infinite combinatorial structures. In *Operations Research '93*, pages 64–67. Physica, Heidelberg, 1994.

[28] S. R. Buss. Alogtime algorithms for tree isomorphism, comparison, and canonization. In *Proc. 5th Kurt Gödel Colloquium on Computational Logic and Proof Theory (KGC)*, volume 1289 of *LNCS*, pages 18–33. Springer, 1997.

[29] S. G. Chaudhuri and K. Mukhopadhyaya. Leader election and gathering for asynchronous fat robots without common chirality. *J. Discrete Algorithms*, 33:171–192, 2015.

[30] W. Chen, C. Wang, and Y. Wang. Scalable influence maximization for prevalent viral marketing in large-scale social networks. *Proceedings of the 16th ACM SIGKDD international conference - KDD '10*, 25(3):1029, 2010.

[31] H. Cheng and G. V. Hadjisophocleous. The modeling of fire spread in buildings by Bayesian network. *Fire Safety Journal*, 44(6):901–908, 2009.

[32] A. L. Christensen. Self-reconfigurable robots—an introduction. (2010, mit press.). *Artificial Life*, 18(2):237–240, 2012.

[33] S. Cicerone. Breaking symmetries on tessellation graphs via asynchronous robots: The line formation problem as a case study. *IEEE Access*, 9:147855–147873, 2021.

[34] S. Cicerone, G. Di Stefano, L. Gąsieniec, and A. Navarra. Asynchronous rendezvous with different maps. In *Structural Information and Communication Complexity*, pages 154–169, Cham, 2019. Springer International Publishing.

[35] S. Cicerone, G. Di Stefano, and A. Navarra. Gathering of robots on meeting-points. In *Proc. 11th Int.'l Symp. on Algorithms and Experiments for Sensor Systems, Wireless Networks and Distributed Robotics (Algosensors)*, volume 9536 of *LNCS*, pages 183–195. Springer, 2015.

[36] S. Cicerone, G. Di Stefano, and A. Navarra. Gathering of robots on meeting-points: feasibility and optimal resolution algorithms. *Distributed Computing*, 31(1):1–50, 2018.

[37] S. Cicerone, G. Di Stefano, and A. Navarra. Asynchronous arbitrary pattern formation: the effects of a rigorous approach. *Distributed Computing*, 32(2):91–132, 2019.

[38] S. Cicerone, G. Di Stefano, and A. Navarra. Asynchronous arbitrary pattern formation: the effects of a rigorous approach. *Distributed Computing*, 32(2):91–132, 2019.

[39] S. Cicerone, G. Di Stefano, and A. Navarra. Asynchronous robots on graphs: Gathering. In *Distributed Computing by Mobile Entities, Current Research in Moving and Computing*, volume 11340 of *LNCS*, pages 184–217. Springer, 2019.

[40] S. Cicerone, G. Di Stefano, and A. Navarra. Embedded pattern formation by asynchronous robots without chirality. *Distributed Computing*, 32(4):291–315, 2019.

[41] S. Cicerone, G. Di Stefano, and A. Navarra. On gathering of semi-synchronous robots in graphs. In *Stabilization, Safety, and Security of Distributed Systems (SSS)*, pages 84–98, Cham, 2019. Springer International Publishing.

[42] S. Cicerone, G. Di Stefano, and A. Navarra. Gathering robots in graphs: The central role of synchronicity. *Theoretical Computer Science*, 849:99–120, 2021.

[43] S. Cicerone, G. Di Stefano, and A. Navarra. "semi-asynchronous": A new scheduler in distributed computing. *IEEE Access*, 9:41540–41557, 2021.

[44] S. Cicerone, G. Di Stefano, and A. Navarra. A structured methodology for designing distributed algorithms for mobile entities. *Information Sciences*, 574:111–132, 2021.

[45] M. Cieliebak, P. Flocchini, G. Prencipe, and N. Santoro. Distributed computing by mobile robots: Gathering. *SIAM J. on Computing*, 41(4):829–879, 2012.

[46] A. Comber, P. Fisher, and R. Wadsworth. What is land cover? *Environment and Planning B: Planning and Design*, 32:199–209, 2005.

[47] M. S. Couceiro, D. Portugal, J. F. Ferreira, and R. P. Rocha. Semfire: Towards a new generation of forestry maintenance multi-robot systems. In *2019 IEEE/SICE International Symposium on System Integration (SII)*, pages 270–276, 2019.

[48] J. Czyzowicz, L. Gasieniec, and A. Pelc. Gathering few fat mobile robots in the plane. *Theoretical Computer Science*, 410(6-7):481–499, 2009.

[49] E. Dahlhaus, D. S. Johnson, C. H. Papadimitriou, P. D. Seymour, and M. Yannakakis. The complexity of multiterminal cuts. *SIAM Journal on Computing*, 23(4):864–894, 1994.

[50] G. D'Angelo, M. D'Emidio, S. Das, A. Navarra, and G. Prencipe. Asynchronous silent programmable matter achieves leader election and compaction. *IEEE Access*, 8:207619–207634, 2020.

[51] G. D'Angelo, M. D'Emidio, S. Das, A. Navarra, and G. Prencipe. Leader election and compaction for asynchronous silent programmable matter. In *Proc. 19th Int.'l Conf. on Autonomous Agents and Multiagent Systems (AAMAS)*, pages 276–284. International Foundation for Autonomous Agents and Multiagent Systems, 2020.

[52] G. D'Angelo, G. Di Stefano, R. Klasing, and A. Navarra. Gathering of robots on anonymous grids and trees without multiplicity detection. *Theoretical Computer Science*, 610:158–168, 2016.

[53] S. Das, P. Flocchini, G. Prencipe, N. Santoro, and M. Yamashita. Autonomous mobile robots with lights. *Theoretical Computer Science*, 609:171–184, 2016.

[54] S. Das, P. Flocchini, N. Santoro, and M. Yamashita. Forming sequences of geometric patterns with oblivious mobile robots. *Distributed Computing*, 28(2):131–145, 2015.

[55] S. Das, R. Focardi, F. L. Luccio, E. Markou, and M. Squarcina. Gathering of robots in a ring with mobile faults. *Theoretical Computer Science*, 764:42–60, 2019.

[56] J. J. Daymude, K. Hinnenthal, A. W. Richa, and C. Scheideler. Computing by programmable particles. In *Distributed Computing by Mobile Entities, Current Research in Moving and Computing*, volume 11340 of *LNCS*, pages 615–681. Springer, 2019.

[57] M. Demange, V. Gabrel, M. A. Haddad, C. Murat, et al. A robust p-center problem under pressure to locate shelters in wildfire context. *EURO Journal on Computational Optimization*, 8(2):103–139, 2020.

[58] M. Demange, M. A. Haddad, and C. Murat. The probabilistic k-center problem. In *Proceedings of the GEOSAFE Workshop on Robust Solutions for Fire Fighting*, pages 62–74, 2018.

[59] M. Demange and C. Tanasescu. A multi-period vertex cover problem and application to fuel management. In *Proceedings of 5th the International Conference on Operations Research and Enterprise Systems (ICORES 2016)*, pages 51–57. SciTePress, 2016.

[60] M. D'Emidio, G. Di Stefano, D. Frigioni, and A. Navarra. Characterizing the computational power of mobile robots on graphs and implications for the euclidean plane. *Information and Computation*, 263:57–74, 2018.

[61] Z. Derakhshandeh, R. Gmyr, T. Strothmann, R. A. Bazzi, A. W. Richa, and C. Scheideler. Leader election and shape formation with self-organizing programmable matter. In *DNA Computing and Molecular Programming - 21st International Conference, 2015*, volume 9211 of *LNCS*, pages 117–132. Springer, 2015.

[62] G. Di Stefano. Mutual visibility in graphs. *Applied Mathematics and Computation*, 419:126850, 2022.

[63] G. Di Stefano and A. Navarra. Gathering of oblivious robots on infinite grids with minimum traveled distance. *Information and Computation*, 254:377–391, 2017.

[64] R. Diestel. *Graph theory*. Springer, 2018.

[65] Y. Dieudonné, F. Petit, and V. Villain. Leader election problem versus pattern formation problem. In *Proc. 24th Int.'l Symp. on Distributed Computing (DISC)*, volume 6343 of *LNCS*, pages 267–281. Springer, 2010.

[66] Directorate Space Security and Migration European Commission Joint Research Centre (EC JRC). CopernicusDEM: Copernicus digital elevation models. Available online: https://doi.org/10.5270/ESA-c5d3d65. Accessed: 2022-5-10.

[67] R. Durrett. *Lecture notes on particle systems and percolation.* Wadsworth & Brooks/Cole Advanced Books & Software, 1988.

[68] J. Díaz and G. Mertzios. Minimum bisection is np-hard on unit disk graphs. *Information and Computation*, 8635, 03 2014.

[69] I. N. Evaggelidis, C. I. Siettos, P. Russo, and L. Russo. Complex network theory criterion to distribute fuel breaks for the hazard control of fire spread in forests. In *AIP Conference Proceedings*, 2015.

[70] J.-B. Filippi, F. Bosseur, and D. Grandi. ForeFire: open-source code for wildland fire spread models. In *ForeFire: open-source code for wildland fire spread models*, pages 275–282. Imprensa da Universidade de Coimbra, Coimbra, 7 2014.

[71] P. Flocchini, G. Prencipe, N. Santoro, and P. Widmayer. Gathering of asynchronous robots with limited visibility. *Theoretical Computer Science*, 337:147–168, 2005.

[72] P. Flocchini, G. Prencipe, N. Santoro, and P. Widmayer. Arbitrary pattern formation by asynchronous, anonymous, oblivious robots. *Theoretical Computer Science*, 407(1-3):412–447, 2008.

[73] P. Flocchini, G. Prencipe, and N. Santoro (Eds.). *Distributed Computing by Mobile Entities, Current Research in Moving and Computing*, volume 11340 of *LNCS*. Springer, 2019.

[74] L. Ford and D. Fulkerson. *Flows in Networks.* Princeton Landmarks in Mathematics and Physics, 1962.

[75] T. Fukuda and Y. Kawauchi. Cellular robotic system (CEBOT) as one of the realization of self-organizing intelligent universal manipulator. In *Proc. of the 1990 IEEE Int.'l Conf. on Robotics and Automation, 1990*, pages 662–667. IEEE, 1990.

[76] H. Galperin and A. Wigderson. Succinct representations of graphs. *Information and Control*, 56(3):183–198, 1983.

[77] M. R. Garey and D. S. Johnson. *Computers and intractability: a guide to the theory of NP-completeness*. W.H. Freeman, 1979.

[78] M. R. Garey, D. S. Johnson, and L. Stockmeyer. Some simplified NP-complete graph problems. *Theoretical Computer Science*, 1(3):237–267, 2 1976.

[79] S. Ghike and K. Mukhopadhyaya. A distributed algorithm for pattern formation by autonomous robots, with no agreement on coordinate compass. In *Proc. 6th Int.'l Conf. on Distributed Computing and Internet Technology, (ICDCIT)*, volume 5966 of *LNCS*, pages 157–169. Springer, 2010.

[80] J. Goldenberg, L. Barak, and E. Muller. Using complex systems analysis to advance marketing theory development: Modeling heterogeneity effects on new product growth through stochastic cellular automata. *Academy of Marketing Science Review*, 9(1), 2001.

[81] S. W. Golomb and D. A. Klarner. Polyominoes. In *Handbook of Discrete and Computational Geometry, 2nd Ed.*, pages 331–352. Chapman and Hall/CRC, 2004.

[82] L. F. Gonzalez, G. A. Montes, E. Puig, S. Johnson, K. Mengersen, and K. J. Gaston. Unmanned aerial vehicles (uavs) and artificial intelligence revolutionizing wildlife monitoring and conservation. *Sensors*, 16(1), 2016.

[83] B. Grünbaum and G. C. Shepard. *Tiling and Patterns*. W. H. Freeman & Co., New York, 1987.

[84] L. J. Guibas, J. E. Hershberger, J. S. Mitchell, and J. S. Snoeyink. Approximating polygons and subdivisions with minimum-link paths. *International Journal of Computational Geometry & Applications*, 3(4):383–415, 1993.

[85] F. Harary and H. Harborth. Extremal animals. *Journal of Combinatorics, Information, & System Sciences*, 1(1):1–8, 1976.

[86] M. S. Innocente and P. Grasso. Swarm of autonomous drones self-organised to fight the spread of wildfires. In *GEOSAFE Workshop on Robust Solutions for Fire Fighting*, 2018.

[87] M. S. Innocente and P. Grasso. Self-organising swarms of firefighting drones: Harnessing the power of collective intelligence in decentralised multi-robot systems. *Journal of Computational Science*, 34:80–101, 2019.

[88] Internet Requests for Comments. The json format. RFC 8259, RFC Editor, 2017, Available online: `https://www.rfc-editor.org/rfc/rfc8259`. Accessed: 2022-5-10.

[89] E. J. Ionascu. Half domination arrangements in regular and semi-regular tessellation type graphs. *Math*, 2012.

[90] F. Jia, K. Zhou, C. Kamhoua, and Y. Vorobeychik. Blocking adversarial influence in social networks. In *Decision and Game Theory for Security*, pages 257–276. Springer International Publishing, 2020.

[91] K. Karra, C. Kontgis, Z. Statman-Weil, J. C. Mazzariello, M. M. Mathis, and S. P. Brumby. Global land use/land cover with sentinel 2 and deep learning. *2021 IEEE International Geoscience and Remote Sensing Symposium IGARSS*, pages 4704–4707, 2021.

[92] G. Karypis and V. Kumar. A fast and high quality multilevel scheme for partitioning irregular graphs. *SIAM Journal on Scientific Computing*, 20(1):359–392, 1998.

[93] D. Kempe, J. Kleinberg, and E. Tardos. Maximizing the spread of influence through a social network. *Theory of Computing*, 11(1):105–147, 2015.

[94] E. Kenler and F. Razzoli. *MariaDB Essentials*. Packt Publishing Ltd, 2015.

[95] E. Khalil, B. Dilkina, and L. Song. CuttingEdge: influence minimization in networks. In *Proceedings of Workshop on Frontiers of Network Analysis: Methods, Models, and Applications at NIPS*, 2013.

[96] Y. Kim, Y. Katayama, and K. Wada. Pairbot: A novel model for autonomous mobile robot systems consisting of paired robots. *CoRR*, abs/2009.14426, 2020.

[97] Y. H. Kim, P. Bettinger, and M. Finney. Spatial optimization of the pattern of fuel management activities and subsequent effects on simulated wildfires. *European Journal of Operational Research*, 197(1):253–265, 8 2009.

[98] M. Kimura, K. Saito, and H. Motoda. Solving the contamination minimization problem on networks for the linear threshold model. In *PRICAI 2008: Trends in Artificial Intelligence*, pages 977–984. Springer Berlin Heidelberg, 2008.

[99] M. Kimura, K. Saito, and H. Motoda. Blocking links to minimize contamination spread in a social network. *ACM Transactions on Knowledge Discovery from Data*, 3(2):1–23, 2009.

[100] H. W. Kuhn and A. W. Tucker. Nonlinear programming. In *Proceedings of the Second Berkeley Symposium on Mathematical Statistics and Probability*, pages 481–492, Berkeley, Calif., 1951. University of California Press.

[101] D. Lichtenstein. Planar formulae and their uses. *SIAM Journal on Computing*, 11(2):329–343, 1982.

[102] Z. Liu, Y. Yamauchi, S. Kijima, and M. Yamashita. Team assembling problem for asynchronous heterogeneous mobile robots. *Theoretical Computer Science*, 721:27–41, 2018.

[103] C. Luo, A. P. Espinosa, D. Pranantha, and A. De Gloria. Multi-robot search and rescue team. In *2011 IEEE International Symposium on Safety, Security, and Rescue Robotics*, pages 296–301, 2011.

[104] T. Maehara, H. Suzuki, and M. Ishihata. Exact computation of influence spread by binary decision diagrams. In *Proceedings of the 26th International Conference on World Wide Web - WWW '17*, pages 947–956. ACM Press, 2017.

[105] H. Mahmoud and A. Chulahwat. Unraveling the Complexity of Wildland Urban Interface Fires. *Scientific Reports*, 8(1):9315, 2018.

[106] D. Martell. The development and implementation of forest and wildland fire management decision support systems: reflections on past practices and emerging needs and challenges. *Mathematical and Computational Forestry & Natural-Resource Sciences (MCFNS)*, 3(1):18–26, 2011.

[107] L. Merino, F. Caballero, J. R. Martínez-de Dios, I. Maza, and A. Ollero. An unmanned aircraft system for automatic forest fire monitoring and measurement. *Journal of Intelligent & Robotic Systems*, 65(1):533–548, Jan 2012.

[108] G. L. Miller and J. H. Reif. Parallel tree contraction, part 2: Further applications. *SIAM Journal on Computing*, 20(6):1128–1147, 1991.

[109] J. P. Minas, J. W. Hearne, and J. W. Handmer. A review of operations research methods applicable to wildfire management. *International Journal of Wildland Fire*, 21(3):189, 5 2012.

[110] J. P. Minas, J. W. Hearne, and D. L. Martell. A spatial optimisation model for multi-period landscape level fuel management to mitigate wildfire impacts. *European Journal of Operational Research*, 232(2):412–422, 2014.

[111] J. Molina and S. Hirai. Aerial pruning mechanism, initial real environment test. *Robotics and Biomimetics*, 4(1):15, Nov 2017.

[112] L. F. P. Oliveira, A. P. Moreira, and M. F. Silva. Advances in forest robotics: A state-of-the-art survey. *Robotics*, 10(2), 2021.

[113] T. B. Paveglio, C. Moseley, M. S. Carroll, D. R. Williams, E. J. Davis, and A. P. Fischer. Categorizing the social context of the wildland urban interface: Adaptive capacity for wildfire and community "archetypes". *Forest Science*, 61(2):298–310, 2015.

[114] Préfet de Corse. Plan de protection des forêts et des espaces naturels. Technical report, 2013, Available online: `http://www.corse-du-sud.gouv.fr/le-ppfeni-en-corse-a373.html`. Accessed: 2021-10-15.

[115] G. Prencipe. Pattern formation. In P. Flocchini, G. Prencipe, and N. Santoro, editors, *Distributed Computing by Mobile Entities, Current Research in Moving and Computing*, volume 11340 of *LNCS*, pages 37–62. Springer, 2019.

[116] Promethee. Forest fires database for mediterranean area in France, 2023, Available online: `https://www.promethee.com/`. Accessed: 2022-5-10.

[117] J. S. Provan. The complexity of reliability computations in planar and acyclic graphs. *SIAM Journal on Computing*, 15(3):694–702, 1986.

[118] A. Pásztor. Gathering simulation of real robot swarm. *Tehnicki vjesnik*, 21(5):1073–1080, 2014.

[119] QGIS Development Team. *QGIS Geographic Information System*. QGIS Association, 2022, Available online: `https://www.qgis.org`. Accessed: 2022-5-10.

[120] R. Rachmawati, M. Ozlen, K. J. Reinke, and J. W. Hearne. A model for solving the prescribed burn planning problem. *SpringerPlus*, 4(1):1–21, 12 2015.

[121] S. Robotics. Colossus. Available online: `https://www.shark-robotics.com/shark-robots`, 2020. Accessed: 2022-11-20.

[122] N. Ruangpayoongsak, H. Roth, and J. Chudoba. Mobile robots for search and rescue. In *IEEE International Safety, Security and Rescue Rototics, Workshop, 2005*, pages 212–217, 2005.

[123] M. Rubenstein, C. Ahler, and R. Nagpal. Kilobot: A low cost scalable robot system for collective behaviors. In *IEEE Int.'l Conf. on Robotics and Automation (ICRA)*, pages 3293–3298. IEEE, 2012.

[124] L. Russo, P. Russo, and C. I. Siettos. A complex network theory approach for the spatial distribution of fire breaks in heterogeneous forest landscapes for the control of wildland fires. *PLOS ONE*, 11(10), 2016.

[125] E. Sahin. Swarm robotics: From sources of inspiration to domains of application. In *Swarm Robotics*, volume 3342 of *LNCS*, pages 10–20. Springer Berlin Heidelberg, 2005.

[126] M. Salman, D. Garzón-Ramos, K. Hasselmann, and M. Birattari. Phormica: Photochromic pheromone release and detection system for stigmergic coordination in robot swarms. *Frontiers Robotics AI*, 7:591402, 2020.

[127] N. Santoro. *Design and Analysis of Distributed Algorithms*. John Wiley & Sons, 2007.

[128] N. Seifter. Properties of graphs with polynomial growth. *Journal of Combinatorial Theory, Series B*, 52(2):222–235, 7 1991.

[129] F. Senra Rivero, J. Venegas Troncoso, C. Ruiz Gutiérrez, F. Castelló Palazón, and A. Seseña Rengel. Revisión de estudio básico de riesgos asociados a los incendios forestales. Technical report, Consejería de Medio Ambiente y Ordenación del Territorio, Andalusia, 2016.

[130] P. Shakarian, A. Bhatnagar, A. Aleali, E. Shaabani, and R. Guo. The independent cascade and linear threshold models. In *SpringerBriefs in Computer Science*, pages 35–48. Springer, 2015.

[131] I. Suzuki and M. Yamashita. Distributed anonymous mobile robots: Formation of geometric patterns. *SIAM Journal on Computing*, 28(4):1347–1363, 1999.

[132] L. Tang and G. Shao. Drone remote sensing for forestry research and practices. *Journal of Forestry Research*, 26(4):791–797, Dec 2015.

[133] H. Tong, B. Prakash, T. Eliassi-Rad, M. Faloutsos, and C. Faloutsos. Gelling, and melting, large graphs by edge manipulation. In *CIKM '12: Proceedings of the 21st ACM international conference on Information and knowledge management*, pages 245–254, 2012.

[134] T. Tucci, B. Piranda, and J. Bourgeois. A distributed self-assembly planning algorithm for modular robots. In *Proc. of the 17th Int.'l Conf. on Autonomous Agents and MultiAgent Systems (AAMAS)*, pages 550–558. ACM, 2018.

[135] M. Turco, S. Jerez, F. J. Doblas-Reyes, A. Aghakouchak, M. C. Llasat, and A. Provenzale. Skilful forecasting of global fire activity using seasonal climate predictions. *Nature Communications*, 9(1):1–9, 2018.

[136] P. Vittorini and F. di Orio. On an Individual-Based Model for Infectious Disease Outbreaks. In *7th International Conference on Practical Applications of Computational Biology & Bioinformatics. Advances in Intelligent Systems and Computing*, volume 222. Springer, Heidelberg, 2013.

[137] P. Vittorini, A. Villani, and F. Di Orio. An individual-based networked model with probabilistic relocation of people and vectors among locations for simulating the spread of infectious diseases. *Journal of Biological Systems*, 18(4), 2010.

[138] J. Whittaker, J. Handmer, and D. Mercer. Vulnerability to bushfires in rural Australia: A case study from East Gippsland, Victoria. *Journal of Rural Studies*, 28(2):161–173, 2012.

[139] Y. Yamauchi and M. Yamashita. Randomized pattern formation algorithm for asynchronous oblivious mobile robots. In *Proc. 28th Int.'l Symp. on Distributed Computing, (DISC)*, volume 8784 of *LNCS*, pages 137–151. Springer, 2014.

[140] L. Yanpei, A. Morgana, and B. Simeone. General theoretical results on rectilinear embedability of graphs. *Acta Mathematicae Applicatae Sinica*, 7:187–192, 1991.

[141] M. Yim, W. Shen, B. Salemi, D. Rus, M. Moll, H. Lipson, E. Klavins, and G. S. Chirikjian. Modular self-reconfigurable robot systems [grand challenges of robotics]. *IEEE Robotics Automation Magazine*, 14(1):43–52, 2007.

[142] H. Zhang, M. A. Alim, M. T. Thai, and H. T. Nguyen. Monitor placement to timely detect misinformation in online social networks. In *2015 IEEE International Conference on Communications (ICC)*, pages 1152–1157, 2015.