Doctoral Thesis

Applied Mathematics

Algorithmic and combinatorial problems on multi-UAV systems

Author:
Luis Evaristo Caraballo de la Cruz

Supervisor:
Dr. José Miguel Díaz Báñez
Dpto de Matemática Aplicada II
Universidad de Sevilla

UNIVERSIDAD D SEVILLA

November 2019

Acknowledgments

This PhD thesis is the result of several years of work and, there have been a bunch of people involved in this difficult undertaking in different ways. I am very grateful to all of them.

Foremost, I would like to express my sincere gratitude to my PhD advisor Prof. José-Miguel Díaz-Báñez for the continuous support of my research, for his patience, motivation, enthusiasm and knowledge. His guidance helped me in all the time of research and writing of this thesis. I could not have imagined having a better advisor and mentor for my PhD study. And, apart of being a great supervisor, he has been helpful and supportive throughout the past years, since I came to Spain.

Several colleagues from different universities and countries have influenced in my research training and, a large part of this thesis is the result of collaborations and discussions with them. I would like to thank all of them for sharing their expertise and advise.

I thank my colleagues at the Department of Applied Mathematics II who have helped me in academic and teaching-related matters. I also thank the administrative staff at the department, IMUS and Vice-Rectorate of Research who were helpful in resolving all of my bureaucratic questions.

Last but not least, I thank my family for their unconditional support despite distance and time. My friends of the "ONU", the former ones and the current ones, all of you are like my Spanish family. And, my girlfriend, for her support, love and care.

The research conducted in the scope of this thesis has received funding from the project GALGO (Spanish Ministry of Economy and Competitiveness, MTM2016-76272-R AEI/FEDER,UE), from the European Union Horizon 2020 research and innovation programme under the Marie Skłodowska-Curie grant agreement No 734922 and from the Spanish Government under the grant agreement FPU14/04705.

Abstract

Mathematics has always been a fundamental piece in robotics and, research in robotics has played an important role in the development of mathematics. This thesis is motivated by the growing interest on problems that appear in aerial robotics applications, specifically, on cooperative systems of multiple aerial robots or drones. Most of the research works in multi-robot systems have focused primarily on construction and validation of working systems, rather than more general and formal analysis of problems and solutions. By contrast, this thesis focuses on formally solving problems of aerial multi-robot systems from a discrete and combinatorial optimization perspective. Inspired on problems of this area, the thesis introduces some new theoretical models and problems of interest for mathematicians and computer scientists.

The following topics are covered in this thesis: (1) *synchronization*: design of a coordination strategy to allow periodical communication between the members of a cooperative team while performing a task along fixed trajectories in a scenario with limited communication range, (2) *robustness*: analysis of the detrimental effects in the performance of a synchronized system when one or more robots fail, (3) *stochastic strategies*: performance analysis of a synchronized system using drones with stochastic decision making, and (4) *task allocation*: decentralized coordination to perform periodical task allocation in order to maintain a balanced work load for all members of a team with limited communication range.

In the first part of the thesis, we study the synchronization problem giving a theoretical characterization of the solutions and, we present an algorithm to build a synchronized system for a given set of covering trajectories. The second part focuses on the study of the robustness in a synchronized system regarding to two key aspects: *covering* of the working area and *communication* between the members of the team. We rigorously study several combinatorial problems to measure how robust a system is to deal with drones failures. Connections of these problems with number theory, graph theory, circulant graphs and polynomial multiplication are shown. The third part is devoted to an analysis of synchronized systems using random aerial robots. This topic is closely related to the random walk theory. It is shown that stochastic strategies increase the robustness of a synchronized system. Finally, this thesis introduces the *block sharing strategy* to address the problem of maintaining a balanced task allocation among the robots by using periodical communications. A proof on the convergence to an optimal task allocation is given and, a case study for structure construction using a cooperative team of aerial robots is presented.

All algorithms developed in this thesis have been implemented and extensive experiments have been conducted to analyze and validate the proposed methods.

Resumen

Las matemáticas siempre han sido una pieza fundamental en el desarrollo de la robótica, así como los problemas de robótica han jugado un importante papel en el desarrollo de las matemáticas. Esta tesis está motivada por el creciente interés en problemas que aparecen en aplicaciones de robótica aérea, específicamente, está enfocada en sistemas cooperativos de múltiples robots aéreos o drones. La mayoría de los trabajos de investigación en sistemas de robots se han centrado en la construcción y validación de arquitecturas desde un enfoque empírico. Por el contrario, esta tesis enfoca el estudio de problemas relacionados con tareas para equipos de robots aéreos desde el punto de vista de la optimización discreta y combinatoria. Inspirada en problemas de este campo, esta memoria plantea nuevos modelos teóricos y problemas de interés para las matemáticas aplicadas y la ciencia computacional.

En esta tesis se abordan los temas siguientes: (1) *sincronización*: diseño de una estrategia de coordinación que permita comunicación periódica entre los miembros de un equipo cooperativo mientras ejecutan una tarea sobre trayectorias fijadas, (2) *robustez*: análisis del efecto que produce el fallo de los agentes en un sistema sincronizado, (3) *estrategias estocásticas*: análisis del funcionamiento de un sistema sincronizado cuando se utilizan drones con toma de decisiones aleatorias, y (4) *asignación de tareas*: coordinación no centralizada usando asignación periódica de tareas que permita mantener una carga de trabajo balanceada.

En la primera parte, se estudia teóricamente el problema de la sincronización, dando condiciones necesarias y suficientes para la existencia de solución y se presenta un algoritmo que construye un sistema sincronizado para un conjunto fijado de trayectorias de vuelo. La segunda parte de la tesis estudia la robustez de un sistema sincronizado teniendo en cuenta dos aspectos fundamentales: el *cubrimiento* del terreno y la *comunicación* entre los miembros del equipo. Se estudian de forma rigurosa problemas combinatorios que surgen cuando se requiere saber cómo de robusto es un sistema con respecto a fallos. Se muestran conexiones con áreas matemáticas como la teoría de números, la teoría de grafos, los grafos circulantes o multiplicación de polinomios. En la tercera parte de la tesis, se estudia la robustez del sistema cuando se introducen decisiones aleatorias de los drones. Se prueba la relación de este problema con la teoría de caminatas aleatorias y se muestra que el uso de estrategias estocásticas supone una mejora de la robustez del sistema sincronizado. Por último, se propone la *estrategia de coordinación por bloques* para la asignación balanceada de tareas. Se prueba la convergencia del método a una asignación óptima y se realiza un estudio de caso para la construcción de una estructura mediante un equipo cooperativo de drones.

Todos los algoritmos desarrollados en esta tesis han sido implementados y se han llevado a cabo diversos experimentos que demuestran la validez de los métodos propuestos.

Table of contents

Chapter 1

Introduction

DRONES or UAVs were synonymous of military applications a few years ago. However, nowadays they are widely applied in several different civil tasks mainly due to their ability to traverse difficult terrains and carry cameras and other sensors to realize monitoring tasks [122]. Additionally, the increasingly low prices of these technologies and their ease of deployment have also boosted their spreading.

Most of the applications of UAVs are based on surveillance and monitoring tasks. For example, traffic surveillance [104, 115], border patrolling [57, 76], crop monitoring [75, 135, 140] and forest fire monitoring [29, 90, 152] to mention just a few of them. We can qualify these applications as 'passive' because the UAVs does not interact with the environment, they just take data using their sensors (i.e., cameras, altitude sensors, thermal cameras, positioning sensors, etc...). However, 'active' applications of UAVs are gaining attention in areas like delivery chains (e.g., Amazon Prime Air project[1] or humanitarian aid delivery [61, 106]), *precision agriculture* [75, 92] and communication networks (e.g., they can be used to establishes wireless communication networks in difficult to access regions or when the communication infrastructure of a region has been damaged by a war or a natural disaster) [91, 138, 154].

Recently, several projects have focused in the design of manipulator aerial platforms [110] to perform, for example, assembly tasks [68]. The Robotic, Vision and Control Group[2](GRVC) of the University of Seville is one of the partners of the H2020 project AEROARMS[3] which is focused in the development of an aerial robotic system equipped with multiple arms and advanced manipulation

[1] `https://www.amazon.com/Amazon-Prime-Air/b?ie=UTF8&node=8037720011`

[2] `https://grvc.us.es/`

[3] `http://aeroarms-project.eu/`

capabilities to be applied in industrial inspection and maintenance [134, 136]. Precisely, from the collaboration between GRVC and our group, the Research Group on **G**eometric **ALGO**rithms & Applications[4](GALGO), the idea of this thesis was born.

1.1 Motivation

Several of the previously presented applications require the solution of many mathematical (sub)problems. For example, path planning tasks appear in every UAVs's application and it can be modeled and solved using resources from geometry, algorithmics and mathematics in general, see a survey of motion planning in [58]. Task allocation is another problem that appears in multi-UAV applications [15, 48, 79]. This problem is related to others that have been previously studied like task allocation on production lines [43, 89] or task allocation on other multi-robot frameworks (e.g., ground robots)[51, 52, 74, 82]. The survey presented in [74] shows several mathematical models for the task allocation problem, one of them is the Multiple Traveling Salesman Problem which has been widely studied for mathematicians and computer scientists [14, 17, 84, 131].

Wireless coverage is another trending application of UAVs [122]. The designing of UAV deployment strategies is one of the challenging topics in this kind of applications [122]. The authors of [122] classify these strategies based on their objective function and cite some of the main works in those categories: minimizing the transmit power of UAVs [93, 119, 96, 117, 112, 9, 97], maximizing the wireless coverage of UAVs [66, 94, 69, 10, 121, 116, 98], minimizing the number of required UAVs to perform a task [70, 118, 120, 155, 86] and maximizing data collection using UAVs [95, 148, 7, 142, 153]. These strategies are related to several mathematical fields like optimization and graph theory and, present very challenging problems in these areas.

Several years ago, the GALGO group, in collaboration with the GRVC, addressed the problem of assigning a velocity profile to each aerial vehicle in real time, such that the separation between them is greater than a given safety distance and the total deviation from the initial planned trajectory is minimized. The results of this work were published in:

> D. Alejo, J.M. Díaz-Báñez, J.A. Cobano, P. Pérez-Lantero, and A. Ollero. "The velocity assignment problem for conflict resolution with multiple aerial vehicles sharing airspace". *Journal of Intelligent & Robotic Systems,* 69 (1-4), 331-346, 2014.

Since then, a great part of GALGO's work is focused on the synergy between aerial robotics and mathematics, and this is precisely the research framework of

[4] `http://alojamientos.us.es/galgo/index.html`

the thesis. The research projects related to multi-robot aerial systems in which we have involved are listed bellow.

Collaborations in the following projects of the GRVC:

1. *Aerial Robotics Cooperative Assembly System* (ARCAS), funded by the European Union (2011-2015).
   https://investigacion.us.es/sisius/proyecto/23363.
   https://cordis.europa.eu/project/rcn/100804.

2. *Cooperative Long Endurance Missions with Aerial Robots*, funded by the Spanish Government (2012-2014).
   https://investigacion.us.es/sisius/sis_proyecto.php?idproy=20140.

3. *MULTIple DRONE Platform for Media Production* (MULTIDRONE), funded by European Union's Horizon 2020 research and innovation programme under grant agreement No 731667 (2017-2019).
   https://investigacion.us.es/sisius/sis_proyecto.php?idproy=28062.
   https://cordis.europa.eu/project/rcn/206392.

And nowadays, with the acquired experience, GALGO group has been granted funding for the project:

4. *Geometric Algorithms for Engineering*, funded by the Spanish Government (2017-2020).
   https://investigacion.us.es/sisius/sis_proyecto.php?idproy=27658.

One of two research lines of this project is focused on the design of geometric and discrete algorithms for multi-robot aerial systems.

Also, we are a partner of the H2020 European project:

5. *Combinatorics of Networks and Computation* (CONNECT), funded by European Union's Horizon 2020 research and innovation programme under the Marie Skłodowska-Curie grant agreement No 734922 (2017-2020).
   https://investigacion.us.es/sisius/sis_proyecto.php?idproy=28049
   https://cordis.europa.eu/project/rcn/207071/factsheet/en.

This project has five work packages and aims to obtain new insights into the behavior of networks, which are studied from a geometric and computational perspective. Our group leads the fourth work package, *Graph-based algorithms for UAVs and for MIR* (https://www.connect-rise.eu/work-package4.php).

1.2 Research lines and objectives

In this thesis we address three main objectives that face algorithmic and combinatorial problems that appear in the field of aerial robotics.

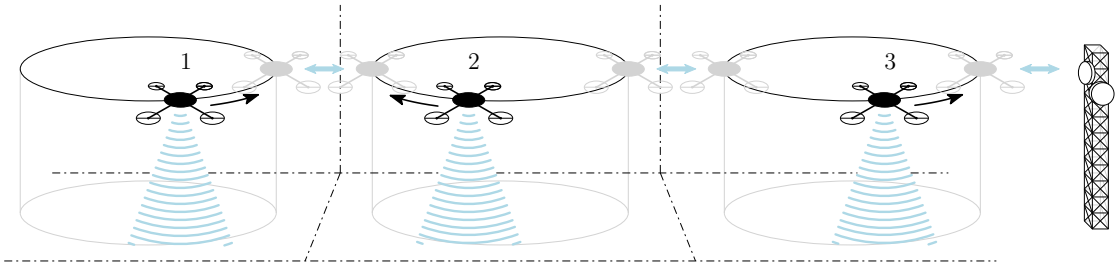


Figure 1.1: Three quadrotors (in black color) traveling along closed trajectories while they are covering pairwise disjoint regions. They can communicate only when they are at the positions marked with gray color. The arrows indicates the travel directions in the regions.

The first part of this thesis is focused on designing a coordination strategy to maintain 'periodical' connectivity between UAVs under communication constraints. Consider the following scenario: we have a big area to monitor with n aerial robots with a very limited communication range. The area is divided into pairwise disjoint regions, one per robot. Each robot monitors its region by traveling periodically along a closed trajectory. Two robots can communicate between them only if they are within the communication range of each other. It is convenient that robots communicate frequently in order to share information about targets or failures of some robot in the system, or simply to report about their mission. In Figure 1.1, robots 1, 2 and 3 are moving in their regions and they can communicate between them only at the positions marked with gray robots. And, robot 3 can communicate with the antenna only if it is at the closest point from its trajectory to the antenna. Thus, the information from robot 1 passes to robot 2, then to robot 3 and, finally it reaches the antenna. In this way, if no robot leaves the system the communication between them is guaranteed and the terrain is completely covered.

The first two objectives of the thesis are related to the scenario presented above:

1. Taking into account the short communication range (with respect to the work area) of the robots, how to coordinate their movement in order to get a connected network between the robots (i.e., a message can travel between any pair of robots)? This is the so-called *synchronization problem* and it is the first objective of the thesis.

2. The second objective is to study the weaknesses of the proposed strategy to solve the synchronization problem and provide robustness mechanisms to recover the network in case that some robots fail (i.e., a robot may need to leaves the system to recharge battery or, it may be knocked down by the enemy in an hypothetical war scenario).

The framework above has several applications apart from surveillance. For

instance, data collection: UAVs are utilized to gather delay-tolerant information from a large number of distributed wireless devices. An example is wireless sensors in precision agriculture applications [122]. Wireless network coverage is another application of the stated framework [122]: a team of UAVs is used as a team of mobile stations to provide wireless communication to a region where there is no communication infrastructure or it has been damaged by a natural disaster or a war.

In the framework above, every UAV has assigned its task. The third objective of the thesis focuses on the design of strategies for task allocation between the members of a team of UAVs:

3. The design of a fully distributed approach, where the robots dynamically allocate their tasks periodically to maintain an optimal allocation according to their capabilities. We consider again that the robots have a limited communication range, so that, permanent communication during the mission is not possible.

1.3 Methodology

The majority of real-world problems are characterized by a high degree of computational complexity due to their complex underlying models which require intelligent exact or heuristic algorithms to achieve high accuracy. Existing approaches often use generic heuristic methods which are adapted to the given scenario without considering the inherent properties of the problem. Such systems often result in low performance and high computational complexity. Consequently, our methodology addresses this knowledge gap and we aim to contribute to the advance in technology by designing problem-specific algorithms. We propose a *bottom-up* methodology where complex problem statements are approximated by a simple model for which we design an efficient algorithm which can be generalized later to real-world scenarios. Moreover, an heuristic method based on an efficient algorithm should be more accurate than a generic heuristic. With this novel methodology we address technological problems in a multi-disciplinary effort to advance in the design of algorithms for performing tasks with drones.

Many problems related to cooperative UAVs rely on geometric networks when a discretization of the problem is considered. For a simple discretized model we can take advantage of using graph or geometric algorithms. The field of Computational Geometry studies fundamental concepts and techniques to solve problems by exploiting its inherent geometry. It is an exciting field that combines clever algorithmic approaches and beautiful geometric structures to obtain general techniques for the efficient storage and processing of spatial data. Geometric algorithms are useful in many application areas such as Operation Research, Robotics, Computer

Graphics, Geographic Information Systems (GIS), Pattern Recognition, and Music Information Retrieval, just to name but a few. These techniques can already be found in decision-support systems in many problems within the fields of facility location, air traffic, data mining, clustering, among others. In all these areas, the geometric problems are solved in a rigorous way, resulting in solutions with guaranteed efficiency, both in terms of memory space and time complexity.

1.4 Related work

At a recent lecture (October 2013) at MAA's Carriage House, Florian Potra of the University of Maryland, Baltimore County, surveyed the rich history of human efforts to create artificial intelligences and self-operating machines (automatas). He said: "While the history of robotics is very short, the idea of a robot may be as old as the earliest mathematical studies. It has always been a special component of the human psyche."[103]. The modern robotics was born in the twentieth century as an interplay between engineering, computer science and mathematics.

All robotic applications requires some level of mathematics, for example, robots with manipulator arms require mathematical models to control their limbs [99]. Path planning for mobile robots requires resources from geometry and optimization to find paths (avoiding obstacles if there are any) of minimum length/cost [80, 58, 149]. In May 2000 there was a meeting at the National Science Foundation in Arlington Virginia on "The Interplay between Mathematics and Robotics" where many leading experts in the United States of America discuss the importance of mathematics in robotics and also the role of robotic problems could play in the development of mathematics (`http://www.math.umbc.edu/~potra/FINALreport.PDF`). As an illustrative example of the synergy between mathematics and robotics we would like to mention the *Dubins path*, which is the shortest curve that connects two points in the two-dimensional Euclidean plane with a constraint on the curvature of the path and with prescribed initial and terminal tangents to the path, and an assumption that the vehicle traveling the path can only travel forward [41]. This is a beautiful and interesting mathematical problem, which is fundamental in the fields of robotics and control theory to plan paths for wheeled robots, aerial robots and underwater vehicles. Another important example of this synergy is Computational Geometry, which recently emerged as a new branch of mathematics and computer science and, has grown mainly motivated by robotics and its applications [111]. To conclude this brief discussion on the interplay between robotics and mathematics, we would like to mention that, nowadays, this relationship has even been studied from the point of view of mathematical education [100, 45, 127].

In the last twenty years, the use of aerial robots and aerial multi-robot systems in monitoring, surveillance, delivery of goods, network coverage, etc. [122], has brought several challenges that have attracted the attention of both, robotics and mathematical research communities. For example, the Vehicle Routing Problem

(VRP) and Traveling Salesman Problem (TSP), are classical problems in the operational research area that have got renewed attention with these applications: see [4] for a survey on VRP instances with applications to multi-objective Unmanned Aerial Vehicle operations and, see [5, 143] for studies on both problems in commercial scenarios that consider a combination of trucks and drones to perform the so-called "last-mile deliver".

However, research in robotics has focused primarily on construction and validation of working systems, rather than more general and formal analysis of problems and solutions [52]. And, most of the formal works, only use mathematics as a resource to build or validate their solutions. This thesis, framed in the synergy between mathematics and robotics, formally studies some problems related to aerial multi-robot systems. Also, to the best of our knowledge, it is a pioneering work that, inspired on problems of this area, introduces some new theoretical models, structures and problems that could be of interest for mathematicians and computer scientists.

The following three subsections show specific related works to each of the three thesis's objectives: the synchronization problem, robustness of synchronized aerial multi-robot systems and task allocation on aerial multi-robot systems.

1.4.1 Previous work related to the synchronization problem

Recall that, in the synchronization problem the robots are performing their tasks in a large area and they have a limited communication range, then they cannot establish a permanent communication network. Then, a coordination strategy is required to guarantee periodic meetings between pairs of robots and, thus allowing the information to be disseminated throughout the team. The coordination of a team of robots under communication constraints has been studied in many specific cases, fundamentally in surveillance and monitoring missions. In this subsection we explore some of the previous works that aims to establish such coordination strategies.

In [76] a cooperative perimeter-surveillance problem is posed and it is offered a decentralized solution that accounts for perimeter growth (expanding or contracting) and insertion/deletion of team members. In this scenario, the team of robots operates on a single trajectory (that can change dynamically) for all the agents. Therefore, two robots moving in opposite directions always meet and can share information with a small communication range, so assigning different directions to the members of the team and changing direction alternately in every meeting event generates the necessary encounters between pairs of neighboring agents. A technique to explore unknown areas using a cooperative team of robots is presented in [47]. Their strategy is based on a data structure called Sensor-based Random Graph (SRG) where the members of the team store environment data and the robots are constantly exploring and updating the SRG. When two robots

fortuitously meet at a point, they share information and generate new collision free motion plans to explore the potential unknown regions. The authors of [56] consider the problem of satisfying communication demands in a multi-agent system where several robots cooperate on a task and a fixed subset of the agents act as mobile routers. The goal is to position the team of robotic routers to provide communication coverage to the remaining client robots. The authors present an adaptive solution to allow for dynamic environments and variable client demands. The main difference with our scenario is that we do not require (or we are not allowed to use) a team of router robots to provide connectivity.

Some of the coordination strategies are based on a convenient subdivision of the area to monitor. For example, in [151], the idea is to divide the area to be explored, a rectangle, for simplicity, into n strips, one for each robot, and then to execute a *lane based search*. The communication links are in the common borders between two consecutive strips. Data flows from the leftmost and rightmost agents to the center agent, and decision control flows from the center to the sides of the team. Based on the form in which the information flows in this strategy they called it *X Synchronization* (XS). This technique is interesting but, unfortunately, can not be applied to solve the problem for general workspaces (no strips possible) with a complex subdivision of the region to monitor. In [64] the following scenario is examined: a mobile network of robots must search, survey, or cover an environment and communication is restricted by relative location. That is, all the members of the team are connected only when they are in specific positions due to obstacles (or something else) in the environment. The idea is to plan the motion of every member of the team to cover the entire region in a minimum of time such that periodically the team is totally connected. They propose an online algorithm that scales linearly in the number of robots and allows for arbitrary periodic connectivity constraints. In [132] an integrated algorithm for task allocation and motion planning is presented. The goal of this algorithm is to keep the connectivity of a cooperative team of robots while their perform a surveillance or exploration mission. The members of the team have limited communication range but the distance between the agents does not admit disconnections. The tasks are assigned to the robots and the motion paths are computed under this restrictions. The topology of the network can changes during the mission but disconnections are not allowed.

Some of the strategies presented above enforce a permanent communication by using extra hardware to provide communication in the team ([56]) or, by allocating tasks and computing paths conveniently in order to maintain a connected network ([132]). Neither of these approaches are feasible in our scenario because as we have posted before we do not have extra hardware to provide communication and, by other hand, moving close enough to maintain the communication may not be a good idea when the region to monitor is large, in these cases it is more convenient that the members of the team are spread out over the region collect-

ing information. Then, a coordination strategy is required to guarantee periodic meetings between pairs of robots aiming to provide a global communication with intermittent information exchange between neighboring robots while they perform their subtasks. The others strategies presented above tackle the communication problem from this perspective, however, they are limited by a fixed topology of the region or a fixed way in which the region is subdivided. In Chapter 2 we aim the problem of establishing a coordination strategy on any complex region using a given subdivision of the terrain.

1.4.2 Related work to robustness of multi-robot aerial systems

The strategy proposed to solve the synchronization problem in Chapter 2 guarantees two important aspects in the proposed scenario:

- *coverage*: the region to monitor is subdivided in n subregions, one per robot, so, every robot covers its assigned region and,
- *communication*: every robot has periodic communications with the robots in neighboring regions.

These periodic intermittent meetings between neighboring robots establish a global communication network allowing the spreading of the information, i.e., it is possible to perform a *broadcast* in the system. If some robot leaves the system due to some catastrophic failure or because it needs to refuel, then the performance of the system is compromised. Taking into account that scalability, fault-tolerance and failure-recovery are important concerns of distributed systems of UAVs, the second objective of the thesis is focused on these matters on the proposed synchronization/coordination strategy. At the end of Chapter 2 the *shifting* protocol is introduced as a recovering strategy to maintain the covering of the region and the communication among the surviving robots when one or more robots leave the system. This protocol does not change the original subdivision of the region, but allows the robots to move from one subregion to another. We experimentally show that shifting protocol is quite robust to random failures of the robots.

In order to determine the robustness of the strategy we use two measures:

- *idle time* (of a subregion): which is the interval of time in which a subregion is not attended by any robot and,
- *broadcast time*: which is the time it takes for a message issued by a robot to reach the full team (of surviving robots).

The idle time is an important quality measure in surveillance and monitoring systems. The frequency of visits as a criterion for measuring the efficiency of

patrolling is called idleness. For a survey of diverse approaches to patrolling based on idleness criteria we refer the reader to [8, 87, 101, 128].

The broadcast time is related to the *meeting time*, defined as the maximum time for two robots to communicate, is another quality measure that is used to evaluate the performance of a system with respect to communication. Collaborative exploration of an unknown environment where the robots meet each other periodically to "cooperate" has been widely considered in robotics, see for example [46, 109]. Broadcast and meeting time have been studied in mobile networks, robotics and random walks, see for example [31, 65, 85, 146].

In Chapter 3, the robustness of our strategy is studied in the worst case. We show that, depending on the subdivision of the region, if certain set of robots fail then some points of the region are not longer visited or the system loses the broadcast ability. It may even happen that one or more robots became *isolated*, i.e., they never meet another robot. This lead us to address the combinatorial problem of determining the minimum number of robots that must fail to break the robustness of the proposed strategy. There are many combinatorial problems motivated by multi-robot applications as it was shown in the first part of this section ([4, 124, 6, 77, 133]). In fact, almost every multi-robot application induces combinatorial optimization problems, for example, determining the minimum of robots to perform the mission, or the minimum cost (travel distance, energy consumption, etc.) to perform a mission with n robots, etc.

To overcome these catastrophic failures of the system in the worst case, in Chapter 4 a randomization of the shifting protocol is introduced. Recently, the study of stochastic multi-robot systems has attracted considerable attention in the field of mobile robots. This approach brings several advantages, such as lower times to complete tasks, cost reduction, higher scalability, and more reliability, among others [150, 44]. In a pure random mobility model, each node randomly selects its direction, speed, and time, independently of other nodes. Some models include random walks, random waypoints or random directions. See [26] for a comprehensive survey.

In Chapter 4, it is shown that our stochastic model generates *random walks*. A random walk on a graph is the process of visiting the nodes of the graph in some sequential random order. The walk starts at some fixed node, and at each step it moves to a neighbor of the current node chosen randomly. There is a vast theoretical literature dealing with random walks. For an overview see e.g. [85, 108]. One of the main reasons that random walk techniques are so appealing for networking applications is their robustness to dynamics. A random walk presents locality, simplicity, low-overhead and robustness to structural changes. Because of these characteristics, applications based on random walks are becoming more and more popular in the networking community. In recent years, different authors have proposed the use of random walks for querying/searching, routing and self-stabilization in wireless networks, peer-to-peer networks, and other distributed

systems [11].

Relevant works in robotics related to the idle-time concept are [87] and [128], where the main objective is to reduce the period between two consecutive visits to any vertex in order to minimize the detection delay for intruders. In the patrolling problem for communication scenarios, some models have been considered in [102] but, the evaluation criterion also is the idle-time. In this thesis, we add to the idle-time two new measures to precisely ensure communication of the team dealing with random strategies: the isolation-time (meeting-time) and the broadcast-time. Recently, the concept of Flying Ad-Hoc Network (FANET), which is basically ad hoc network between UAVs, has been introduced in [13]. FANETs can be seen as a subset of the well-known mobile ad hoc networks (MANETs), where the use of random walks has been intensive [59]. Thus, an opportunity to extend the stochastic methods to UAVs has been opened. Our contribution is to provide an example of using random strategies as an alternative to existing protocols for UAVs networks.

1.4.3 Related work to task allocation on multi-robot systems

The third objective of the thesis focuses on designing of an strategy to maintain a balanced task allocation to face dynamical changes of the system or the environment. The task allocation problem has been widely studied in different areas. Centralized mechanisms have advantages including a guaranteed optimal solution. However, their limitations are well known, i.e. their slow response to dynamic events and vulnerability to failures. In dynamic and uncertain environments, decentralized mechanisms are preferred. Decentralized task allocation approaches for autonomous agents have been well studied in robotics [39, 52, 74, 78, 81]. Reference [50] presents a dynamic task allocation scheme using an auction process for a heterogeneous robot team. Reference [38] introduces a market-based multi-robot task allocation architecture in which robots are modeled as self-interested agents with the goal of maximizing individual profits. In [22, 30] two algorithms on market-based decision strategy are proposed: the consensus-based auction algorithm (CBAA) and its generalization to the multi-assignment problem, the consensus-based bundle algorithm (CBBA). In [19], the CBBA is extended taking into account obstacle regions in order to generate collision free paths for UAVs.

In practice, robots have a limited communication capability which may influence development and performance of the task allocation algorithm significantly. In [40] an interception scenario involving a team of UAVs and a group of moving targets is considered. Their problem is to coordinate the team of UAVs based on their local information using a decentralized task allocation approach using intermittent and asynchronous communications. In [2, 1], the authors focus on a decentralized algorithm to ensure information propagation in area monitoring

missions with a fleet of heterogeneous UAVs with limited communication range. The strategy follows a distributed one-to-one coordination schema and has been adopted to maintain a balanced area partition among the robots.

The task allocation problem is also related to the well-known assembly line balancing (ALB) optimization problem in operational research [21, 113]. The ALB problem deals with partitioning the total assembly operations into a set of m elementary tasks $o_i (i = 1, \ldots, m)$ with times t_i, and assigning them to a team of n assembly robots $r_k (k = 1, \ldots, n)$ such that all robots approximately spend equal assembly times and the so-called "precedence constraints" between operations are satisfied. Assuming that the set S_k of tasks is assigned to the k-th robot, its assembly time is $t(S_k) = \sum_{j \in S_k} t_j$. The ALB problem is NP-hard [113] and soft computing approaches have been proposed in [107]. The main goal in an ALB model is the allocation of the tasks among stations so that the precedence relations are not violated and a given objective function is optimized. Besides balancing a newly designed assembly line, an existing assembly line has to be re-balanced periodically or after certain changes in the production process or the production plan.

1.5 Contributions

In this section the contributions of this thesis are listed on an item basis. The first objective of the thesis is addressed in Chapter 2. In here we present:

1. A *synchronized system* as a strategy to coordinate the movement of the robots in their regions in such a way that pairs of robots are *synchronized* and arrive periodically at the same time to a region where they are close enough to exchange information.

We also present:

2. A fail-recovery approach on a synchronized system that allows robots to leave the system or be inserted into it, minimizing detrimental effects on system performance.

In this chapter we also define the *communication graph* of a synchronized system as a graph whose vertices are the trajectories and there is an edge connecting two trajectories if their robots can exchange information periodically.

A synchronized system has three very important properties that assure the good performance of the system: the robots have meetings periodically, all the points in the trajectories are visited periodically and it has the ability for messages broadcasting between the robots in the system. And, if some robots fail then the surviving ones apply the fail-recovery approach trying to maintain these properties. However, if a large enough number of robots fail, the system may lose some of these

properties. In Chapter 3 we focus on the study of the *resilience* of a synchronized system, that is, what is the maximum number of robots that can fail such that the system maintains these properties? Actually, we study the problem from the other point of view, what is the minimum number ρ such that there exist ρ robots whose removal causes the loss of some of these properties? In Chapter 3 these questions are addressed.

3. Three resilience measures are presented:
   (a) *coverage resilience* as the minimum number ρ_c such that, there exist ρ_c robots whose removal causes that some point is no longer covered,
   (b) *k-isolation resilience* as the minimum number ρ_i such that, there exist ρ_i robots whose removal causes that at least k surviving robots have no longer any meeting and,
   (c) *broadcasting resilience* as the minimum number ρ_b such that, there exist ρ_b robots whose removal causes that it is no longer possible to make a broadcast in the system.

Obviously, the larger the resilience, the more fault tolerant the system is. These measures lead us to three combinatorial problems: given a synchronized system what are the values of these measures? In [23] the notion of k-isolation resilience is introduced and an $O(n^2)$ time algorithm for computing the 1-resilience is proposed. Chapter 3 studies these problems and shows that:

4. The problem of computing any of the presented measures depends on the topology of the communication graph. Let n denote the number of trajectories in the synchronized system. The following table summarizes our results:

	Communication graph's topology			
Resilience measures	Tree	Cycle	$N \times M$ Grid*	General
coverage	$O(1)$	$O(n)$	$O(T_{gcd}(N,M))^{\dagger}$	$O(n)$
k-isolation	NP-Hard	$O(n)$	$O(1)$	NP-Hard
broadcasting	$O(n^{3/2})$	$O(n)$	$O(1)$	$O(n\kappa \cdot \min\{\kappa^3 + n, n\kappa\})^{\ddagger}$

* We say that the communication graph has an $N \times M$ grid topology if the trajectories are arranged forming a grid of N rows and M columns, notice that $n = N * M$.
† $T_{gcd}(N,M)$ denotes the required time to compute $\gcd(N,M)$. Taking into account that $n = N * M$ then, $T_{gcd}(N,M) = O(\log n(\log\log n)^2 \log\log\log n)$ according to [129].
‡ κ denotes the value of the broadcasting resilience.

5. We also analyze the problem of computing k-isolation resilience for specific values of k: an almost linear time (in the number of trajectories) algorithm is proposed to compute the 1-isolation resilience for any synchronized system (improving the $O(n^2)$ time algorithm presented in [23]). And, if the communication graph is a tree then the 1-isolation resilience can be computed in linear time, the 2-isolation resilience can be computed in $O(\min(t^2, n \log n))$ time and the k-isolation resilience ($k > 2$) can be computed in $O(tn^{k-1})$ time, where $\sqrt{\pi n}/2 - 1 \leq t \leq n - 1$ is a parameter that depends on the topology of the communication tree.

After analyzing the weaknesses and strengths of a synchronized system, in Chapter 4:

6. Two random variants synchronized systems are presented where the robots apply the fail-recovery approach (item 2 of this list) randomly. We study the behavior of the expected meeting time, expected broadcast time and expected idle time.

In Chapter 5:

7. The block-information-sharing (BIS) strategy is formally presented as a generalization of the one-to-one approach presented in [1, 2] and the convergence of the BIS strategy is formally proved in a general task allocation scenario.

8. Moreover, it is shown how to use this strategy to design a fault-tolerant approach for structure construction using a cooperative team of aerial robots. The robots work in parallel and the dynamic assignments of the tasks are performed using blocks in order to maintain a balanced allocation where each aerial robot approximately spends the same time to construct the assigned section. Thus, the maximum time a robot spends to complete the assigned section of the construction is minimized.

1.5.1 Related publications and collaborations

The main contributions of this thesis are based on the following publications, which are published or currently under review in peer-reviewed journals and conferences.

- J. Díaz-Báñez, L. E. Caraballo, M. A. Lopez, S. Bereg, I. Maza, and A. Ollero. "A general framework for synchronizing a team of robots under communication constraints". *IEEE Transactions on Robotics, 33(3):748–755, June 2017.* (Chapter 2)

- J. Díaz-Báñez, E. Caraballo, M. A. Lopez, S. Bereg, I. Maza, and A. Ollero. "The synchronization problem for information exchange between aerial robots under communication constraints". *In 2015 IEEE International Conference on Robotics and Automation (ICRA), pages 4650–4655, May 2015.* (Chapter 2)

- S. Bereg, L. E. Caraballo, J. M. Díaz-Báñez, and M. A. Lopez. "Computing the k-resilience of a synchronized multi-robot system". *Journal of Combinatorial Optimization, 36(2):365–391, Aug 2018.* (Chapter 3)

- S. Bereg, L. E. Caraballo, J. M. Díaz-Báñez, and M. A. Lopez. "On the robustness of a synchronized multi-robot system". Submitted to *Journal of Combinatorial Optimization.* (Chapter 3)

- S. Bereg, L. E. Caraballo, J. M. Díaz-Báñez, and M. Lopez. "The uncovering-resilience of a synchronized multi-robot system". *In XVII Spanish Meeting on Computational Geometry. Alicante, Spain, Jun 2017.* (Chapter 3)

- S. Bereg, L. E. Caraballo, J. M. Díaz-Báñez, and M. Lopez. "Computing the k-resilience of a synchronized multi-robot system". *In European Workshop on Computational Geometry (EuroCG2017). Malmö, Sweden, Apr 2017.* (Chapter 3)

- L. E. Caraballo, J. M. Díaz-Báñez, R. Fabila-Monroy, and C. Hidalgo-Toscano. "Patrolling a terrain with cooperrative UAVs using random walks". *In 2019 International Conference on Unmanned Aircraft Systems (ICUAS), pages 828–837, June 2019.* (Chapter 4)

- L. E. Caraballo, J. M. Díaz-Báñez, I. Maza, and A. Ollero. "The block-information-sharing strategy for task allocation: A case study for structure assembly with aerial robots". *European Journal of Operational Research, 260(2):725–738, 2017.* (Chapter 5)

- L. E. Caraballo, J. J. Acevedo, J. M. Díaz-Báñez, B. C. Arrue, I. Maza, and A. Ollero. "The block-sharing strategy for area monitoring missions using a decentralized multi-uav system". *In 2014 International Conference on Unmanned Aircraft Systems (ICUAS), pages 602–610, May 2014.* (Chapter 5)

The following publications are related to, or inspired by, the work presented in this thesis, but are not considered in this manuscript.

- L. E. Caraballo, J. M. Díaz-Báñez, R. Fabila-Monroy, A. Fernández, and F. Rodríguez. "Collision avoidance for aerial vehicles using turn-angles". *In XVIII Spanish Meeting on Computational Geometry. Girona, Spain, Jul 2019.*

- S. Bereg, L. E. Caraballo, and J. M. Díaz-Báñez. "Efficient inspection of underground galleries using k robots with limited energy". *In ROBOT 2017: Third Iberian Robotics Conference, pages 706–717. Springer International Publishing, 2018. ISBN: 978-3-319-70833-1.*

1.5.2 Thesis outline

Chapter 2 addresses the synchronization problem (first objective of the thesis) and, presents the *synchronized systems* as a solution for this problem. Chapters 3 and 4 are devoted to the second objective: robustness of synchronized systems. The first of these chapters studies some drawbacks that can appear in a synchronized system when a set of robots fail and, it studies several measures to determine how fault-tolerant a system is. This chapter contains most of the theoretical work of the thesis. In Chapter 4 we introduce some stochastic behavior in the drones of a synchronized system in order to overcome the drawbacks shown in the previous chapter. The third and last objective, task allocation, is addressed in Chapter 5, where the block-information-sharing strategy is proposed as a general approach to maintain a balanced task allocation on multi-robot system under communication constraints. Finally, the work is concluded in Chapter 6, where future work directions are discussed.

Chapter 2

The synchronization problem

COMMUNICATION is required in a cooperative robotic system that performs a task in a decentralized manner and the robustness of the system depends on the reliability of the communication links between the agents involved. In many scenarios, direct communication among all the agents cannot be guaranteed (large workspace with respect to the communication range of the system's members, for instance). In such scenarios, the communication links are available only when neighboring robots reside within a short communication range [25, 125]. This is the situation assumed in this chapter.

Let us consider a team of robots which are periodically traveling along predetermined closed simple trajectories (i.e., smooth Jordan curves) while performing an assigned task (see examples in Section 1.2). Each of the agents needs to communicate information about its operation to other agents, but the communication interfaces have a limited range. Hence, when two agents are within the communication range, a communication link is established, and the information is exchanged. We say that two agents are "synchronized" if they can exchange information periodically. In this chapter, the following synchronization problem is considered: given the path geometries of a group of robots, schedule the motion of the robots along their trajectories so that the number of synchronized pairs of robots is maximized.

In order to illustrate some of the issues arising in the synchronization problem, let us consider the situation shown in Figure 2.1 with trajectories P_1, P_2 and P_3, and three robots moving in the same direction (counterclockwise) along their respective trajectories. Suppose that, due to limited communication, there is only a possible communication link between every pair of trajectories (a possible commu-

nication link is indicated by a pair of connected red hollow points in Figure 2.1). How can we guarantee that each pair of robots is synchronized, i.e., each pair of robots arrives at a possible communication link between their trajectories at the same time periodically? Let x, y and z be the required time to travel (as it's shown in Figure 2.1) between the link positions of P_1, P_2 and P_3, respectively. It is easy to see that if the three agents spent $x + y + z$ time to make a tour in their trajectories, then every pair of robots is synchronized and they exchange information every tour. The video at `https://youtu.be/4Bo590VcKbY` shows an animation of a simple system with circular trajectories where every pair of neighboring robots is synchronized.

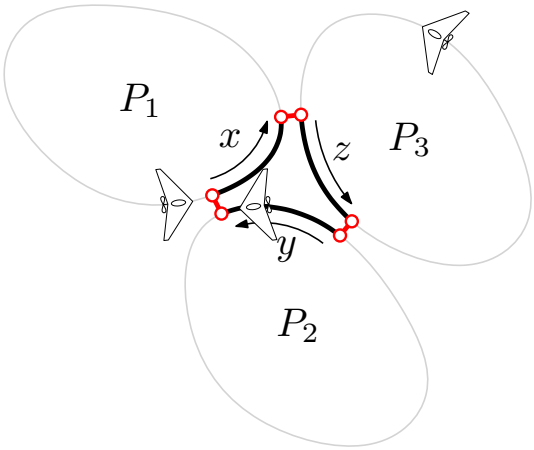


Figure 2.1: Simple scenario of the synchronization problem. Two robots can exchange information only if they are on a pair of connected red hollow points.

The synchronization problem arises naturally in missions of surveillance or monitoring [2, 102], wireless network coverage [66, 24], in structure assembly while the robots are loading and placing parts in a structure [18], or even in the exploration process looking for parts to be assembled as performed in the ARCAS project (`http://www.arcas-project.eu`), to name but a few applications. In fact, its eventual solution may find many applications beyond the initial problems posed here.

Moreover, notice that the synchronization problem is even more relevant in aerial robots due to:

- Rotorcraft robots (e.g., helicopters or multirotor systems) have very demanding energy requirements that limit the flight endurance. Then, hovering, which is very energy demanding, waiting for other aerial robots to communicate or to perform cooperatively a task, should be minimized by means of synchronization.

- The synchronization of fixed wing aircraft imposes more strict constraints when they should meet to interchange information due to the velocity of

these aircrafts that may lead to communication losses when using short range communication devices.

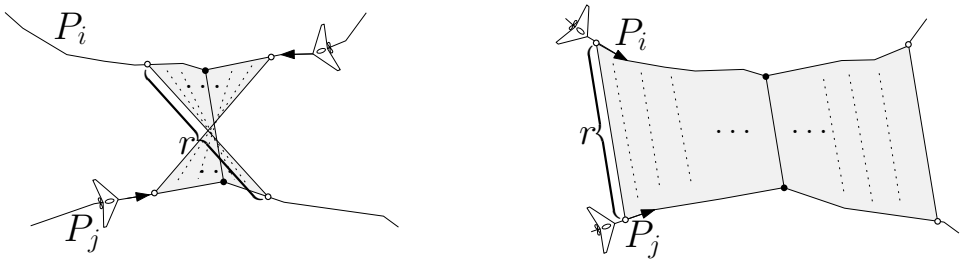
Additionally, the short communication range between aerial robots, and unmanned aerial vehicles in general, is interesting from the point of view of security. In fact, this short range communication may avoid communication jamming, which is a significant threat in the practical application of unmanned aerial vehicles [139].

2.1 Problem formulation

The ingredients of the general problem considered here are the following:

- A team $V_1, V_2, \ldots, V_n$ of n aerial robots need to share information while cooperating in the execution of a task in a decentralized way.
- Each vehicle V_i has a fixed communication range r_i and flies along a specified closed trajectory P_i. The routes are disjoint, eliminating concerns about collisions.
- A potential communication link exists between the trajectories of vehicles V_i and V_j if and only if the minimum distance between the trajectories does not exceed the value $\min\{r_i, r_j\}$. The two vehicles may exchange information at all times when the distance between them is less than or equal to $\min\{r_i, r_j\}$.
- We say that two robots are *neighbors* if there exists a potential communication link between their trajectories; and two neighbors are *synchronized* if they visit that link at the same time.
- Given two synchronized robots V_i and V_j, the *communication region* of V_i and V_j, is given by the two connected arcs R_{ij} and R_{ji} on P_i and P_j respectively such that when V_i flies on R_{ij} and V_j flies on R_{ji}, the distance between V_i and V_j is kept within $\min\{r_i, r_j\}$. Two cases arise, depending on whether the two neighbors fly in opposite directions, one clockwise (CW) and the other counter-clockwise (CCW), or in the same direction. Figure 2.2a shows the latter case while Figure 2.2b illustrates the former. Notice that the two cases affect differently the length of the arcs R_{ij} and R_{ji}. We will later see that they also impact differently the robustness of the system.

Problem 2.1.1 (The Synchronization Problem). *Given a set of n robots, each performing part of a cooperative task within a closed trajectory, and exchanging information with a fixed communication range, schedule the trajectories such that the number of synchronized robot pairs is maximized. When this number equals the number of communication links, the system is fully synchronized.*



(a) One direction (b) Two directions

Figure 2.2: Representation of the communication region between two robots V_i and V_j (gray region): (a) two neighboring robots moving in clockwise direction and (b) two neighboring robots moving in opposite directions. Notice that both figures use the same communication range ($r = min\{r_i, r_j\}$).

First, observe that under a general model, even the synchronization between two robots cannot be guaranteed. Consider, for example, a system consisting of only two robots, each moving at constant speed with a very small communication range. Then, if the ratio of the two trajectory lengths is not rational, a synchronized motion is not possible. Typically, the methods used in the literature achieve synchronization by changing the speeds of the robots by small amounts, e.g., they allow for the possibility of one vehicle “waiting" for the other. Unfortunately, this simple approach is only feasible for two vehicles. For a team of cooperative robots, a more delicate theoretical study is required.

The following questions need to be answered in order to implement an efficient and robust multi-robot coordinated system: (i) When can a cooperative multi-robot system be synchronized without changing the robot speeds? and (ii) in case of a robot failure, can the schedules be slightly altered in order to complete the global task in a new synchronized system?

This chapter aims to answer the above two questions. Based on theoretical results on a simplified model, we propose an algorithm that is efficient and robust in the face of catastrophic robot failures where the synchronization of a large team of robots is assured.

2.2 Theoretical results in a simplified model

The general methodology we are proposing is to first obtain strong results on a simplified, albeit not entirely practical model, and then to adapt the newly acquired theoretical knowledge to more general and realistic models. Accordingly, we discuss how to extend the approach for the simple model to others models in Section 2.3 where we consider, for example, heterogeneous vehicles, non-circular routes, etc.

Let us consider a simplified model for which the basic results can be stated.

In this simplified model all aerial robots fly at the same altitude and move along equal and pairwise disjoint circular trajectories at the same speed with the same communication range. Without loss of generality, assume that a robot spent one unit of time to make a tour in a trajectory.

Let $T = \{C_1, C_2, \ldots, C_n\}$ be the set of pairwise disjoint unit circles representing the flight trajectories of the n robots and let $\epsilon < 0.5$ be the communication range of the robots. Let C_i and C_j be two trajectories such that the distance between their centers is less than or equal to $2 + \epsilon$. Then, the robots in C_i and C_j can potentially share information (see Figure 2.3).

In order to model the ensuing communication constraints we define the *graph of potential links* of the system *with respect to range* ϵ as follows:

Definition 2.2.1 (Graph of potential links)**.** *Let $T = \{C_1, \ldots, C_n\}$ be a set of trajectories and let ϵ be the communication range of the robots. The* graph of potential links *of the system* with respect to range ϵ *is a planar graph $G_\epsilon(T) = (V, E_\epsilon)$ whose vertices are the circle centers and whose edges connect two centers if their distance is less than or equal to $2 + \epsilon$ (see Figure 2.4). Moreover, if two trajectories, C_i and C_j, are connected by an edge of E_ϵ then they are* neighbors *and, the edge connecting them is denoted by (i, j).*

Remark 2.2.2. *In the simplified model, the graph of potential links is actually the* intersection graph *of a set of enlarged trajectories, namely, the set of disks of radius $1 + \epsilon/2$ with the same trajectory centers. Therefore, the graph of potential links can be computed in linear time [16].*

Since communication is an important issue in cooperative scenarios, this work focuses on sets of trajectories whose graphs of potential links is connected. Assume for the rest of the chapter that we are working with a given set of trajectories T and a fixed communication range ϵ such that the graph of potential links $G_\epsilon(T)$ is connected.

In the analysis that follows, it will be convenient to denote a point of a trajectory by the angle that it forms with horizontal positive axis measured in counterclockwise direction, i.e., the angle in polar coordinates (see Figure 2.3).

Since the robots move at constant speed, it suffices to know the starting position and movement direction of a robot in order to compute its position at any time. Thus, we can define a *schedule* of a system as the set of starting positions and travel directions on every trajectory.

Definition 2.2.3 (Schedule)**.** *Let $T = \{C_1, \ldots, C_n\}$ be a set of trajectories. A* schedule *on T is a pair of functions (α, δ), $\alpha : T \to [0, 2\pi)$ and $\delta : T \to \{-1, 1\}$, where $\alpha(C_i)$ is the starting position in the circle C_i and $\delta(C_i)$ is the movement direction in the circle C_i, 1 corresponds to counterclockwise direction (CCW) and -1 to clockwise direction (CW). At an arbitrary time t, a robot's position in circle C_i is:*

$$\alpha(C_i) + 2\pi \cdot \delta(C_i) \cdot t. \tag{2.1}$$

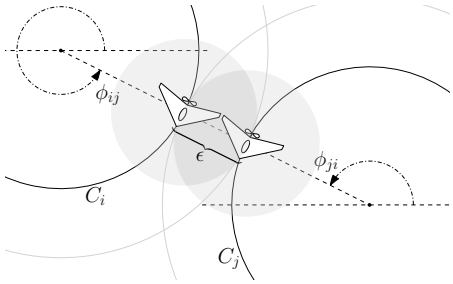


Figure 2.3: The simple model. Robots in C_i and C_j can share information because the distance between them is less than or equal to ϵ. Notice that they are at the link positions ϕ_{ij} and ϕ_{ji}, respectively.

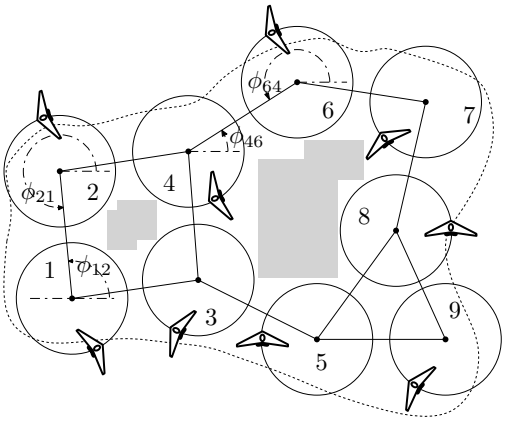


Figure 2.4: Representation of a set $C_1, C_2, \ldots, C_9$ of unit circles and the underlying graph of potential links.

Let (i, j) be an edge in E_ϵ. For the sake of simplicity we consider that two robots in trajectories C_i and C_j can establish a communication link if and only if they are at the same time over the segment that connects the centers of C_i and C_j, see Figure 2.3. The following definition describes formally these positions.

Definition 2.2.4 (Link position). *Let $T = \{C_1, \ldots, C_n\}$ and ϵ be the set of trajectories and the communication range of a system, respectively. Let $G_\epsilon(T) = (V, E_\epsilon)$ be the graph of potential links of the system. Let (i, j) be an edge in E_ϵ. The* link position of C_i with respect to C_j, *denoted by ϕ_{ij}, is the angle corresponding to the point of C_i closest to C_j. Clearly, if ϕ_{ij} is defined, so is ϕ_{ji} and $\phi_{ji} = \pi + \phi_{ij}$ (see Figure 2.3).*

Using the previous two definitions we can extend the notion of synchronization to trajectories in the following way:

Definition 2.2.5 (Synchronized trajectories). *Let $S = (\alpha, \delta)$ be a schedule on a system whose set of trajectories is T and whose communication range is ϵ. Two trajectories C_i and C_j of T are* synchronized by S *if there exists a $0 \leq t < 1$ such that:*

$$\alpha(C_i) + 2\pi \cdot \delta(C_i) \cdot t = \phi_{ij} \quad \textit{and} \quad \alpha(C_j) + 2\pi \cdot \delta(C_j) \cdot t = \phi_{ji}.$$

Notice that, two robots in a synchronized pairs of trajectories, are synchronized between them and they "meet each other" (establish a link position) every time unit (see Figure 2.3).

Definition 2.2.6 (Communication graph). *Let S be a schedule on a system whose set of trajectories is T and whose communication range is ϵ. The* communication graph *of S, is a* spanning subgraph[1] *of $G_\epsilon(T) = (V, E_\epsilon)$ whose edges are the pairs of synchronized trajectories.*

Notice that, if the communication graph of a schedule is connected then a message can be relay between every pair of robots on the system (may be passing through multiple robots). We are interested in this kind of schedules.

Definition 2.2.7 (Synchronization schedule). *Let S be a schedule on a system whose set of trajectories is T and whose communication range is ϵ. Let G be the communication graph of S. If G is connected then we say that S is a* synchronization schedule. *If $G = G_\epsilon(T)$ then S is a* full synchronization schedule.

We say that a system is *synchronized* if the team of robots (one per trajectory) is using a synchronization schedule.

In the following we are going to characterize the relation between pairs of synchronized trajectories. In order to save notation we will use α_i and δ_i instead of $\alpha(C_i)$ and $\delta(C_i)$ to refer the starting position and travel direction in a trajectory C_i, respectively.

[1] A *spanning subgraph* of a graph H is another graph whose set of vertices is exactly the same set of vertices of H and whose set of edges is a subset of the set of edges of H.

Lemma 2.2.8. *Let G be the communication graph of an schedule $S = (\alpha, \delta)$. Let (i, j) be an edge of G. If $\delta_j = \delta_i$ then $\alpha_j = \pi + \alpha_i$.*

Proof. There exists a value t such that:

$$\alpha_i + 2\pi\delta_i t = \phi_{ij} \text{ and } \alpha_j + 2\pi\delta_j t = \phi_{ji}.$$

By using Definition 2.2.4 we get that:

$$\alpha_j + 2\pi\delta_j t = \alpha_i + 2\pi\delta_i t + \pi.$$

Using that $\delta_i = \delta_j$ the result follows. □

Lemma 2.2.9. *Let G be the communication graph of an schedule $S = (\alpha, \delta)$. Let (i, j) be an edge of G. If $\delta_i = -\delta_j$ then $\alpha_j = \phi_{ji} + \phi_{ij} - \alpha_i$.*

Proof. There exists a value $t \geq 0$ such that:

$$\phi_{ij} = \alpha_i + 2\pi\delta_i t \text{ and } \phi_{ji} = \alpha_j + 2\pi\delta_j t.$$

Then:

$$\phi_{ij} - \alpha_i = 2\pi\delta_i t \text{ and } \phi_{ji} - \alpha_j = 2\pi\delta_j t.$$

Using that $\delta_i = -\delta_j$ the result follows. □

Let T be set of trajectories and let ϵ be the communication range of the robots. Let $S = (\alpha, \delta)$ be a schedule on T and, let $S' = (\alpha', \delta)$ be another schedule on T using the same directions in every trajectory. Consider that $\alpha'(C_i) = \alpha(C_i) + 2\pi\varepsilon\delta(C_i)$ for all $C_i \in T$ where ε is a positive constant. Note that, in this case the starting positions of S' are the positions of S after ε units of time. Therefore, if the system starts with a set of n robots using the schedule S, after ε units of time, the robots will be moving like if they have just started using the schedule S'. Then, we can say that S and S' are almost the same schedule, they are *equivalent*. Notice that S and S' have the same communication graph.

Definition 2.2.10 (Equivalent schedules)**.** *Let T be set of trajectories and let ϵ be the communication range of the robots. Two schedules $S = (\alpha, \delta)$ and $S' = (\alpha', \delta')$ on T are* equivalent *if for all $t \geq 0$ there exists a value $t' = t'(t) \geq 0$ such that:*

$$\alpha'_i + 2\pi\delta'_i t' = \alpha_i + 2\pi\delta_i t \text{ for all } 1 \leq i \leq n.$$

More informally, this definition says that if we have two imaginary sets of robots following schedules S and S', respectively, and we take a picture of the team who is following schedule S at some arbitrary time, then, S and S' are equivalent if we can get exactly the same picture at some instant of time in the team following S'.

Remark 2.2.11. *Let $\mathcal{S}$ be the set of all the possible schedules on a system. It is easy to prove that the previous definition establishes an* equivalence relation *between the elements of $\mathcal{S}$. That is, for any schedules S, S' and S'' in $\mathcal{S}$:*

- *S is equivalent to S* (reflexive property)*,*
- *if S is equivalent to S' then S' is equivalent to S* (symmetric property)*, and*
- *if S is equivalent to S' and S' is equivalent to S'' then S is equivalent to S''* (transitive property)*.*

Then, this equivalence relation provides a partition of $\mathcal{S}$ into (disjoint) equivalence classes*.*

The following theorem characterizes the equivalent schedules.

Theorem 2.2.12. *Two schedules $S = (\alpha, \delta)$ and $S' = (\alpha', \delta')$ are equivalent if and only if $\delta' = \pm\delta$ and $\alpha' = \alpha + c\delta$ where $0 \leq c < 2\pi$ is a constant.*

Proof. Firstly, we prove that if S and S' are equivalent, then $\delta' = \pm\delta$ and $\alpha' = \alpha + c\delta$ where $c \in \mathbb{R}$ is a constant. We will start by showing that for all $1 \leq i \leq n$, if $\delta'_1 = \delta_1$ then $\delta'_i = \delta_i$ and, if $\delta'_1 = -\delta_1$ then $\delta'_i = -\delta_i$. After that, we are going to prove that if $\alpha'_1 = \alpha_1 + c\delta_1$ then $\alpha'_i = \alpha_i + c\delta_i$.

Let $p_i(t) = \alpha_i + 2\pi\delta_i t$ and $p'_i(t) = \alpha'_i + 2\pi\delta'_i t$ denote the position in trajectory C_i at time t using schedules S and S', respectively. Let $t > 0$ be an arbitrary instant of time. Let $t' > 0$ be the minimum value such that $p_1(t) = p'_1(t')$. Since S and S' are equivalent we deduce that $p_i(t) = p'_i(t')$ for all $1 \leq i \leq n$. Let $0 < \varepsilon < 1/2$. Notice that:

$$p_i(t + \varepsilon) = \alpha_i + 2\pi\delta_i(t + \varepsilon) = p_i(t) + 2\pi\delta_i\varepsilon.$$

Suppose that $\delta'_1 = \delta_1$, then $p'_1(t' + \varepsilon) = p_1(t + \varepsilon)$. Since S and S' are equivalent, then, for all $1 \leq i \leq n$:

$$\begin{aligned} p'_i(t' + \varepsilon) &= p_i(t + \varepsilon) \\ p'_i(t') + 2\pi\delta'_i\varepsilon &= p_i(t) + 2\pi\delta_i\varepsilon. \end{aligned}$$

Using that $p'_i(t') = p_i(t)$ and $0 < 2\pi\varepsilon < \pi$ we deduce that $\delta'_i = \delta_i$ for all $1 \leq i \leq n$. Now, suppose that $\delta'_1 = -\delta_1$, then $p'_1(t' + 1 - \varepsilon) = p_1(t + \varepsilon)$. Since S and S' are equivalent, then, for all $1 \leq i \leq n$:

$$\begin{aligned} p'_i(t' + 1 - \varepsilon) &= p_i(t + \varepsilon) \\ p'_i(t') - 2\pi\delta'_i\varepsilon &= p_i(t) + 2\pi\delta_i\varepsilon. \end{aligned}$$

Using that $p'_i(t') = p_i(t)$ and $0 < 2\pi\varepsilon < \pi$, we deduce that $\delta'_i = -\delta_i$ for all $1 \leq i \leq n$.

Let $0 \leq c < 2\pi$ be a constant such that $\alpha'_1 = \alpha_1 + c\delta_1$. For all times t, t' such that $p'_i(t') = p_i(t)$ we have that:

$$\begin{aligned} p'_i(t') &= p_i(t) \\ \alpha'_i + 2\pi\delta'_i t' &= \alpha_i + 2\pi\delta_i t \\ \alpha'_i &= \alpha_i + 2\pi\delta_i t - 2\pi\delta'_i t' \\ \alpha'_i &= \alpha_i + 2\pi\delta_i(t - \delta_i\delta'_i t'). \qquad (\delta_i^2 = 1) \end{aligned}$$

Notice that the last equation can be rewritten as:

$$\alpha'_i = \alpha_i + 2\pi\delta_i \cdot \texttt{frac}(t - \delta_i\delta'_i t')$$

where $\texttt{frac}(t - \delta_i\delta'_i t')$ denotes the fractional part of $(t - \delta_i\delta'_i t')$. Note that the integer part of $(t - \delta_i\delta'_i t')$ corresponds to entire loops in a trajectory because a robot spends one time unit in a tour, thus, it doesn't affect the position in a circle. For $i = 1$, regardless the values of t and t', the value $2\pi \cdot \texttt{frac}(t - \delta_1\delta'_1 t')$ is the constant c. Notice that, if $\delta' = \pm\delta$ then $\delta'_i\delta_i = \delta'_1\delta_1$ for all $1 \leq i \leq n$. Thus, $2\pi \cdot \texttt{frac}(t - \delta_1\delta'_1 t') = c$ for all $1 \leq i \leq n$. This completes the proof of the first implication.

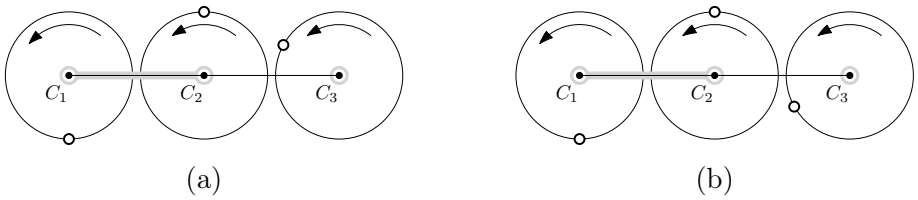
Now, let us focus on proving that if $\delta' = \pm\delta$ and $\alpha' = \alpha + c\delta$ $(0 \leq c < 2\pi)$ then $S = (\alpha, \delta)$ and $S' = (\alpha', \delta')$ are equivalent schedules. Let t be an arbitrary instant of time and let t' be the minimum non-negative value of the form:

$$t' = s\left(t - \frac{c}{2\pi}\right) + k \quad \text{where } k \in \mathbb{Z} \text{ and } s = \frac{\delta}{\delta'} = \begin{cases} 1 & \text{if } \delta' = \delta \\ -1 & \text{if } \delta' = -\delta \end{cases}$$

For all trajectory $C_i \in T$ the position in C_i at time t' following schedule S' is:

$$\begin{aligned} p'_i(t') &= \alpha'_i + 2\pi\delta'_i t' \\ &= \alpha'_i + 2\pi\delta'_i\left(s\left(t - \frac{c}{2\pi}\right) + k\right) \\ &= \alpha'_i + 2\pi\delta'_i s\left(t - \frac{c}{2\pi}\right) + 2\pi\delta'_i k \\ &= \alpha'_i + 2\pi\delta'_i\frac{\delta_i}{\delta'_i}\left(t - \frac{c}{2\pi}\right) && (2\pi\delta'_i k \text{ doesn't affect the position}) \\ &= \alpha'_i + 2\pi\delta_i t - c\delta_i \\ &= \alpha_i + c\delta_i + 2\pi\delta_i t - c\delta_i && (\alpha'_i = \alpha_i + c\delta_i \text{ by hypothesis}) \\ &= \alpha_i + 2\pi\delta_i t \\ &= p_i(t). \end{aligned}$$

Then S and S' are equivalent. □



(a) (b)

Figure 2.5: (a) and (b) represent the same system with different and non-equivalent schedules $S = (\alpha, \delta)$ and $S' = (\alpha', \delta')$, respectively. S and S' have the same communication graph $G = (V = \{1, 2, 3\}, E = \{(1, 2)\})$ (which is drawn using light-gray color) and $\delta' = \delta$ (the directed arcs represent the travel direction in every trajectory). The hollow points represent the used starting positions.

Let T and ϵ be the set of trajectories and the communication range of a system of robots respectively. Let $G_\epsilon = (V, E_\epsilon)$ be the graph of potential links of the system. Let $S = (\alpha, \delta)$ be a schedule of the system. Let $V|S = \{A, B\}$ denote the partition of V induced by δ, i.e., $\delta_i = \delta_j$ for all $(i, j) \in (A \times A) \cup (B \times B)$ and $\delta_i = -\delta_j$ for all $(i, j) \in (A \times B)$. Let $\mathcal{C}$ be an equivalent class of schedules on the system. From the previous theorem it is easy to see that $\mathcal{C}$ determines a bipartition $\{A, B\}$ of V such that, for all schedules S and S' of $\mathcal{C}$ we have $V|S = V|S' = \{A, B\}$.

Notice that any bipartition $\{A, B\}$ of V and any function of starting points $\alpha : T \longrightarrow [0, 2\pi)$ determines an equivalence class of schedules on the system.

Lemma 2.2.13. *If two schedules are equivalent then they have the same communication graph.*

Proof. Let $S = (\alpha, \delta)$ and $S' = (\alpha', \delta')$ be two equivalent schedules with communication graphs $G = (V, E)$ and $G' = (V, E')$, respectively. Let $(i, j) \in E$ be an edge of G. Then, at some time $t \geq 0$

$$\alpha_i + 2\pi\delta_i t = \phi_{ij} \text{ and } \alpha_j + 2\pi\delta_j t = \phi_{ji}.$$

By using the definition of equivalent schedules, we know there exists an instant $t' \geq 0$ such that:

$$\alpha'_i + 2\pi\delta'_i t' = \phi_{ij} \text{ and } \alpha'_j + 2\pi\delta'_j t' = \phi_{ji}.$$

Then, $(i, j) \in E'$ (by using the definition of communication graph). Thus, $E \subseteq E'$. Applying the same argument for every edge of G' we get that $E' \subseteq E$, then $G = G'$. □

Notice that the reverse implication of the previous lemma is not true. See Figure 2.5, where two schedules are shown with the same communication graph and robots moving in CCW direction in both cases. However, the shown schedules are not equivalent.

Definition 2.2.14. *Let $G_\epsilon = (V, E_\epsilon)$ be the graph of potential links of a system. Let S be a schedule of the system. Let $G = (V, E)$ be a subgraph of G_ϵ. We say that G is* synchronizable *by S if G is a subgraph of the communication graph of S (i.e., C_i is synchronized with C_j for all $(i, j) \in E$). We also say that S* synchronizes *G.*

From the previous definition and Lemma 2.2.13, the following result is deduced:

Corollary 2.2.15. *Let S and S' be two equivalent schedules of a system whose graph of potential links is G_ϵ. If a subgraph G of G_ϵ is synchronizable by S then G is also synchronizable by S'.*

Lemma 2.2.16. *Let $G_\epsilon = (V, E_\epsilon)$ be the graph of potential links of a system. Let G be a spanning subgraph of G_ϵ. Let S and S' be two schedules that synchronize G. If $V|S = V|S'$ and G is connected then S and S' are equivalent synchronization schedules.*

Proof. Let $S = (\alpha, \delta)$ and $S' = (\alpha', \delta')$. Notice that if G is connected then the communication graph of S and S' are also connected, thus, S and S' are synchronization schedules. Since $V|S = V|S'$, we get that $\delta' = \pm\delta$. Therefore, for proving that S and S' are equivalent we only need to demonstrate that $\alpha' = \alpha + c\delta$ for some $c \in [0, 2\pi)$ (Theorem 2.2.12). Let $c \in [0, 2\pi)$ be a constant such that $\alpha'_1 = \alpha_1 + c\delta_1$. Let us prove that $\alpha'_i = \alpha_i + c\delta_i$ for all $i \in V$ by using induction on the length of the shortest path from 1 to i in G (recall that G is connected). Let $V_{1,d} \subseteq V$ be the set of vertices of G at distance d or less from 1. Let us start with $d = 0$. Then $V_{1,0} = \{1\}$ and $\alpha'_i = \alpha_i + c\delta_i$ for all $i \in V_{1,0}$.

Assume as inductive hypothesis that for a fixed value $d \geq 0$ $\alpha'_i = \alpha_i + c\delta_i$ for all $i \in V_{1,d}$.

Let j be an arbitrary vertex in $V_{1,d+1} \setminus V_{1,d}$, then, a shortest path P in G from 1 to j has length $d+1$. Let i be the penultimate vertex in P, i.e., $P = [1, \ldots, i, j]$. Notice that $i \in V_{1,d}$.

If $\delta_j = \delta_i$ then $\delta'_j = \delta'_i$ (because $\delta' = \pm\delta$) and:

$$\begin{aligned} \alpha'_j &= \alpha'_i + \pi && \text{(by Lemma 2.2.8)} \\ &= \alpha_i + c\delta_i + \pi && \text{(by inductive hypothesis)} \\ &= \alpha_j + c\delta_j. && \text{(by Lemma 2.2.8 and using } \delta_i = \delta_j) \end{aligned}$$

If $\delta_j = -\delta_i$ then $\delta'_j = -\delta'_i$ (because $\delta' = \pm\delta$) and:

$$\begin{aligned} \alpha'_j &= \phi_{ij} + \phi_{ji} - \alpha'_i && \text{(by Lemma 2.2.9)} \\ &= \phi_{ij} + \phi_{ji} - \alpha_i - c\delta_i && \text{(by inductive hypothesis)} \\ &= \alpha_j + c\delta_j. && \text{(by Lemma 2.2.9 and using } \delta_i = -\delta_j) \end{aligned}$$

Then, by using Theorem 2.2.12, S and S' are equivalent. □

From Theorem 2.2.12, Lemma 2.2.16 and Lemma 2.2.13 we deduce the following result that characterizes equivalent synchronization schedules.

Theorem 2.2.17. *Let $S = (\alpha, \delta)$ and $S' = (\alpha', \delta')$ be two schedules with communication graphs G and G', respectively. S and S' are equivalent synchronization schedules if and only if G and G' are the same connected graph and $\delta' = \pm\delta$.*

Remark 2.2.18. *From Theorem 2.2.17 it is deduced that an equivalence class $\mathcal{C}$ of synchronization schedules is determined by a graph $G = (V, E)$ and a bipartition $\{A, B\}$ of V such that:*

- *G is the communication graph of all the schedules in $\mathcal{C}$ and,*
- *$\{A, B\}$ is the bipartition determined by all the schedules in $\mathcal{C}$.*

Moreover, if a schedule S is an optimum solution for Problem 2.1.1 then every schedule S' equivalent to S is an optimum solution for Problem 2.1.1 as well. Therefore, the solution for Problem 2.1.1 is not single schedule, rather it is an equivalence class of schedules.

The following result states that the schedule which is the solution of the synchronization problem (Problem 2.1.1) is a synchronization schedule, that is, its communication graph is connected.

Lemma 2.2.19. *Let $G_\epsilon = (V, E_\epsilon)$ be the graph of potential links of a system. If a schedule S reaches the maximum possible number of synchronized pairs of trajectories then S is a synchronization schedule.*

Proof. Let $S = (\alpha, \delta)$. Let $G = (V, E)$ be the communication graph of S. For the sake of contradiction, assume that G is not connected. Let $(x, y) \in E_\epsilon \setminus E$ such that x and y are not connected by any path in G. Take a bipartition $\{A, B\}$ of V such that $x \in A$ and $y \in B$. Think in $S_A = \left(\alpha^{(A)}, \delta^{(A)}\right)$ and $S_B = \left(\alpha^{(B)}, \delta^{(B)}\right)$ as two schedules resultant from the restriction of S to A and B, respectively. Take the schedules $S'_A = \left(\alpha'^{(A)}, \delta^{(A)}\right)$ and $S'_B = \left(\alpha'^{(B)}, \delta^{(B)}\right)$ equivalent to S_A and S_B such that $\alpha'^{(A)}_x = \phi_{xy}$ and $\alpha'^{(B)}_y = \phi_{yx}$, respectively. Let $S' = \left(\alpha'^{(A)} \cup \alpha'^{(B)}, \delta\right)$. The number of connected components of the communication graph $G' = (V, E')$ of S' is less than the number of connected components of G. Notice that, for every edge (i, j) of G, (i, j) also belongs to G' and, G' additionally contains the edge (x, y) which is not in G. So, $|E'| > |E|$, this is a contradiction. □

In the following subsection we focus on the study of an algorithm to build a synchronization schedule S for a system (not necessarily with maximum number of synchronized pair of robots). Notice that, using this schedule a message can be relay between every pair of robots on the system.

2.2.1 The synchronization algorithm

Let $G_\epsilon = (V, E_\epsilon)$ be the graph of potential links of a system. Given a connected spanning subgraph G of G_ϵ and a partition $\{A, B\}$ of V, the algorithm that we propose in this subsection computes a synchronization schedule S such that S synchronizes some spanning tree of G and $V|S = \{A, B\}$.

The idea of the algorithm is to build a sequence $\{1\} = Z_0 \subset Z_1 \subset \cdots \subset Z_{|V|-1} = V$ of nested subsets of V. The algorithm builds this sequence by stages $i = 0, 1, \ldots, (|V| - 1)$ in this order. When $i = 0$ the algorithm builds $Z_0 = \{1\}$ and assigns the values $\alpha_1 = 0$ and $\delta_1 = 1$ if $1 \in A$ and $\delta_1 = -1$ if $1 \notin A$ (that is $1 \in B = V \setminus A$). To pass from stage i to stage $i + 1$ the algorithm look for a vertex w in $V \setminus Z_i$ adjacent to another vertex $v \in Z_i$. Then $Z_{i+1} = Z_i \cup \{w\}$ and the algorithm computes the values of α_w and δ_w using the following rule:

$$\begin{aligned} \alpha_w &= \begin{cases} \alpha_v + \pi & \text{if } (v, w) \in (A \times A) \cup (B \times B), \\ \phi_{vw} + \phi_{wv} - \alpha_v & \text{otherwise,} \end{cases} \\ \delta_w &= \begin{cases} \delta_v & \text{if } (v, w) \in (A \times A) \cup (B \times B), \\ -\delta_v & \text{otherwise.} \end{cases} \end{aligned} \tag{2.2}$$

Property 2.2.20. *Let $G = (V, E)$ be the input graph. Let e_i denote the used edge to pass from Z_i to Z_{i+1}. For all $0 \leq i < |V|$ the graph $T_i = (Z_i, \{e_0, e_1, \ldots, e_{i-1}\})$ is a subtree of G, the schedule $S = (\alpha, \delta)$ synchronizes T_i and, $Z_i|S = \{A \cap Z_i, B \cap Z_i\}$.*

Proof. Let us prove it by induction on i. For $i = 0$, the tree $T_0 = (\{1\}, \emptyset)$ is a single node, so, it is a subtree of G. At this stage the algorithm has assigned the values $\alpha_1 = 0$ and $\delta_1 = 1$ if $1 \in A$ and $\delta_1 = -1$ if $1 \in B$. Obviously, S synchronizes T_0 and $Z_0|S = \{A \cap Z_0, B \cap Z_0\}$ because T_0 is formed by the single vertex 1.

Suppose as inductive hypothesis that the property holds for a fixed value i. Let (v, w) the used edge to pass from Z_i to Z_{i+1}. Recall that $Z_{i+1} = Z_i \cup \{w\}$, $v \in Z_i$ and $w \in V \setminus Z_i$. The graph T_{i+1} is the resultant graph of connecting the new vertex w to T_i by using the edge (v, w), using that T_i is a tree (inductive hypothesis) we get that T_{i+1} is a tree as well. Let (x, y) be an edge of T_{i+1}. If $(x, y) \neq (v, w)$ then the trajectories C_x and C_y are synchronized by S (inductive hypothesis). When passing from stage i to $i + 1$, the algorithm computes the values of α_w and δ_w by using the already computed values of α_v and δ_v (see equation 2.2). By using Lemma 2.2.8 and Lemma 2.2.9 we can check that the trajectories C_v and C_w are synchronized by S as well. Thus, S synchronizes T_{i+1}. Additionally: if v and w are in the same set of the partition $\{A, B\}$ (i.e., $(v, w) \in (A \times A) \cup (B \times B)$) then $\delta_w = \delta v$ (see equation 2.2) and, w and v will be in the same set of the partition $Z_{i+1}|S$. If v and w are in different sets of the partition $\{A, B\}$ (i.e., $(v, w) \in (A \times B)$) then $\delta_w = -\delta v$ (see equation 2.2) and, w and v will be in different sets of the partition $Z_{i+1}|S$. In both cases, using that $Z_i|S = \{A \cap Z_i, B \cap Z_i\}$ (inductive hypothesis) we get that $Z_{i+1}|S = \{A \cap Z_{i+1}, B \cap Z_{i+1}\}$. The result follows. □

Algorithm 1: `SynchronizationAlgorithm`

Input : $G = (V, E)$, A
Output: α, δ

$\phi \leftarrow$ `link_positions`(G)
$Q \leftarrow$ `queue`$()$
`enqueue`$(Q, 1)$
$M \leftarrow$ [**false** , . . . , **false**] `// list of length` $|V|$
$M[1] \leftarrow$ **true**
$\alpha \leftarrow [0, \ldots, 0]$ `// list of length` $|V|$
$\delta \leftarrow [0, \ldots, 0]$ `// list of length` $|V|$
$\alpha[1] \leftarrow 0$
$\delta[1] \leftarrow -1$
if 1 *in* A **then**
  $\delta[1] \leftarrow 1$
end
while Q *is* ***not*** *empty* **do**
  $v \leftarrow$ `dequeue`(Q)
  $l \leftarrow$ `neighbors`(G, v)
  foreach w *in* l **do**
    if ***not*** $M[w]$ **then**
      if *(*v *in* A ***and*** w *in* A*)*
       or *(*v ***not*** *in* A ***and*** w ***not*** *in* A*)* **then**
        $\alpha[w] \leftarrow \alpha[v] + \pi$
        $\delta[w] \leftarrow \delta[v]$
      else
        $\alpha[w] \leftarrow \phi[v, w] + \phi[w, v] - \alpha[v]$
        $\delta[w] \leftarrow -\delta[v]$
      end
      $M[w] \leftarrow$ **true**
      `enqueue`(Q, w)
    end
  end
end
return α, δ

Algorithm 1 implements the previous idea using a breadth-first search approach starting at vertex 1. The input is a connected spanning graph $G = (V, E)$ of the system and the set $A \subseteq V$ (notice that $B = V \setminus A$). The output of the algorithm are the functions α and δ of a synchronization schedule S. Notice that the sequence $\{1\} = Z_0 \subset Z_1 \subset \cdots \subset Z_{|V|-1} = V$ is emulated by using the Boolean array M. If M has k elements marked as **true** then, these k elements conform the set Z_{k-1}. The constructed breadth-first tree is the spanning tree $T = (Z_{|V|-1}, \{e_0, \ldots, e_{|V|-2}\})$ of G. Then, from Property 2.2.20 the following result follows:

Corollary 2.2.21. *Let $G = (V, E)$ be a connected spanning graph of a system. Let $\{A, B\}$ be a partition of V. Let S be the output schedule of* `Synchronization Algorithm`$(G, \{A, B\})$. *Then, $V|S = \{A, B\}$ and S synchronizes a spanning tree of G.*

Lemma 2.2.22. *Algorithm 1 takes $O(n)$ time where n is the number of trajectories in the system.*

Proof. Let $G = (V, E)$ be the input graph of the algorithm. It is easy to check that this algorithm takes $O(|E|)$ time (every edge is analyzed twice, one per incident vertex line 13 and 14). Taking into account that G_ϵ is a planar graph and G is a subgraph of G_ϵ we get that $|E| \in O(n)$ where n is the number of trajectories in the system. Thus, the algorithm takes $O(n)$ time. □

Theorem 2.2.23. *Let $G = (V, E)$ be a connected spanning graph of a system. Let $\{A, B\}$ be a partition of V. If there is a schedule S that synchronizes G and $V|S = \{A, B\}$, then, the output schedule S' of* `SynchronizationAlgorithm`$(G, \{A, B\})$ *is equivalent to S. Moreover, S' synchronizes G as well.*

Proof. From Corollary 2.2.21 we have that S' synchronizes some spanning tree T of G. This tree T is also synchronized by S (because T is a subgraph of G) then, by using Lemma 2.2.16, S and S' are equivalent and they have the same communication graph (Lemma 2.2.13). Therefore, S' synchronizes G. □

Definition 2.2.24. *Let $G_\epsilon = (V, E_\epsilon)$ be the graph of potential links of a system. Let $G = (V, E)$ be a subgraph of G_ϵ. We say that G is* synchronizable *by a bipartition $\{A, B\}$ of V if there is a schedule S that synchronizes G and $V|S = \{A, B\}$. We also say that S* verifies *that G is synchronizable by $\{A, B\}$.*

Let $G_\epsilon = (V, E_\epsilon)$ be the graph of potential links of a system. Let $G = (V, E)$ be a connected spanning subgraph of G_ϵ. Notice that any two schedules S and S' that verify that G is *synchronizable* by a bipartition $\{A, B\}$ of V are in the same equivalence class of synchronization schedules.

From Corollary 2.2.21 the following result is derived:

Lemma 2.2.25. *Let $G_\epsilon = (V, E_\epsilon)$ be the graph of potential links of a system. Any spanning tree T of G_ϵ is synchronizable by any bipartition $\{A, B\}$ of V. Moreover, the output schedule of* `SynchronizationAlgorithm`$(T, \{A, B\})$ *verifies that T is synchronizable by $\{A, B\}$.*

Let $G_\epsilon = (V, E_\epsilon)$ be the graph of potential links of a system. And let G be a connected spanning subgraph of G_ϵ. In subsection 2.2.2 we are going to study the conditions G must fulfills to be synchronizable by the bipartition $\{V, \emptyset\}$. Subsection 2.2.3 focuses on the study of the conditions that G must fulfills to be synchronizable by the bipartition $\{A, B\}$ considering that G is bipartite with partite sets A and B.

2.2.2 Robots moving in the same direction

Suppose that we have a synchronized system where all the robots are moving in the same direction. Let S be the synchronization schedule that the robots are following. Let $G = (V, E)$ be the communication graph of S. Then, G is synchronizable by the partition $\{V, \emptyset\}$ of V. In this subsection we focus on the study of graphs synchronizable by the partition $\{V, \emptyset\}$.

Lemma 2.2.26. *Let $G = (V, E)$ be a connected spanning graph of a system. Let $S = (\alpha, \delta)$ be a schedule that synchronizes G such that $V|S = \{V, \emptyset\}$. If $(i, j) \in E$ and $(i, k) \in E$ then $\alpha_j = \alpha_k$.*

Proof. From Lemma 2.2.8 we have that $\alpha_i = \alpha_j + \pi$ and $\alpha_k = \alpha_i + \pi$. Then, $\alpha_k = (\alpha_j + \pi) + \pi = \alpha_j + 2\pi = \alpha_j$. □

Using Lemma 2.2.26 we derive the following corollary:

Corollary 2.2.27. *Let $G = (V, E)$ be a connected spanning graph of a system. If G is synchronizable by $\{V, \emptyset\}$ then G does not contain cycles of odd length.*

Proof. Let $S = (\alpha, \delta)$ be a schedule that verifies that G is synchronizable by $\{V, \emptyset\}$. For the sake of contradiction, suppose that G contains and odd cycle $\langle i_0, i_1, \ldots, i_{2c}, i_0 \rangle$. From the previous lemma we know that $\alpha_{i_0} = \alpha_{i_2} = \cdots = \alpha_{i_{2c}}$. And by Lemma 2.2.8 we know that $\alpha_{i_0} = \alpha_{i_{2c}} + \pi$ and this is a contradiction. □

Now we are ready to prove the main result for the simplified model when the robots move in the same direction.

Theorem 2.2.28. *Let $G = (V, E)$ be a connected spanning graph of a system. G is synchronizable by $\{V, \emptyset\}$ if and only if G is bipartite. Moreover, the output schedule S of* `SynchronizationAlgorithm`(G, V) *verifies that G is synchronizable by $\{V, \emptyset\}$.*

Proof. From Corollary 2.2.27 we deduce that if G is synchronizable by $\{V, \emptyset\}$ then G is bipartite.

Let us focus on proving that if G is bipartite then it is synchronizable by $\{V, \emptyset\}$. Let $S = (\alpha, \delta)$ be output schedule of `SynchronizationAlgorithm`(G, V). Notice that $\delta_i = 1$ for all $i \in V$, then $V|S = \{V, \emptyset\}$. So, we only need to prove that S synchronizes G. There is a spanning tree T of G such that S synchronizes T (Corollary 2.2.21). Notice that $\alpha_i = 0$ or $\alpha_i = \pi$ for all $i \in V$. Let (i, j) be an edge of G which is not in T. Let $P = [i = v_1, \ldots, v_{2c} = j]$ be the path from i to j in T. Notice that P has an even number of vertices because G is bipartite. Thus, if $\alpha_{v_1} = 0$ then $\alpha_{v_2} = \pi$, $\alpha_{v_3} = 0$, $\ldots$, $\alpha_{v_{2c}} = \pi$ and, if $\alpha_{v_1} = \pi$ then $\alpha_{v_2} = 0$, $\alpha_{v_3} = \pi$, $\ldots$, $\alpha_{v_{2c}} = 0$. In any case $\alpha_j = \alpha_i + \pi$, then trajectories C_i and C_j are synchronized by Lemma 2.2.8. The result follows. □

From this theorem, the following result is directly derived:

Corollary 2.2.29. *Let T be a set of trajectories and let ϵ be the communication range of the robots. The synchronized system with robots moving in the same direction that ensures maximum number of synchronized pairs of robots is determined by the bipartite subgraph of G_ϵ with maximum number of edges.*

We can compute the graph G_ϵ of potential links in linear time (Remark 2.2.2), then, we can test the bipartiness of G_ϵ in linear time. If it is bipartite we are done, else a reasonable strategy is to use a bipartite subgraph with the maximum number of edges. Finding the maximum bipartite subgraph of a general graph is an NP-complete problem [72]. However, such a subgraph can be found in polynomial time when the input graph is planar [60]. Alternatively, there exist various approximation algorithms that can be useful in practice (see for instance [42]).

The synchronization problem (Problem 2.1.1), with the restriction that all robots move in the same direction, can be solved using Algorithm 2. Notice that, with this algorithm, we obtain a synchronization schedule which is a representative of an equivalence class of schedules that are optimum solutions for this problem.

2.2.3 Robots moving in opposite directions

Let $G_\epsilon = (V, E_\epsilon)$ be the graph of potential links of a system. In this subsection we derive conditions that ensure the synchronization of a system where every pair of synchronized robots are moving in opposite directions. Note that such a model can only be deployed on a bipartite communication graph $G = (V, E)$. In this case, the partition $V|S$ determined by the used synchronization schedule S corresponds exactly to the partite sets of the bipartite communication graph (i.e., $V = A \cup B$, $A \cap B = \emptyset$, $E \subseteq (A \times B) \cap E_\epsilon$ and $V|S = \{A, B\}$).

Algorithm 2: Computing a synchronization schedule where all the robots move in the same direction.

Input : T, ϵ
Output: α, δ
$G_\epsilon \leftarrow$ `computeIntersectionGraph`$(T, 1+\epsilon/2)$
if G_ϵ *is bipartite* **then**
  $G \leftarrow G_\epsilon$
else
  $G \leftarrow$ `maximumBipartiteSubgraph`(G_ϵ)
end
return `SynchronizationAlgorithm`$(G, V(G))$ `// Algorithm 1`

Theorem 2.2.30. *Let $G = (\{A, B\}, E)$ be a connected bipartite spanning graph of a system. G is synchronizable by $\{A, B\}$ if and only if every cycle $\langle i_1, \ldots, i_{2c}, i_1\rangle$ in G satisfies:*

$$(\phi_{i_1 i_2} + \phi_{i_2 i_1}) - (\phi_{i_2 i_3} + \phi_{i_3 i_2}) + \cdots + (\phi_{i_{2c-1} i_{2c}} + \phi_{i_{2c} i_{2c-1}}) - (\phi_{i_{2c} i_1} + \phi_{i_1 i_{2c}}) \in 2\pi\mathbb{Z}.$$

Moreover, the output schedule S of `SynchronizationAlgorithm`$(G, \{A, B\})$ *verifies that G is synchronizable by $\{A, B\}$.*

Proof. Let $S = (\alpha, \delta)$ be a schedule that verifies that G is synchronizable by $\{A, B\}$. Let $\langle i_1, \ldots, i_{2c}, i_1\rangle$ be an arbitrary cycle in G. From Lemma 2.2.9 we have that:

$$\begin{aligned}
\alpha_{i_2} &= (\phi_{i_1 i_2} + \phi_{i_2 i_1}) - \alpha_{i_1} \\
\alpha_{i_3} &= (\phi_{i_2 i_3} + \phi_{i_3 i_2}) - (\phi_{i_1 i_2} + \phi_{i_2 i_1}) + \alpha_{i_1} \\
&\vdots \quad \vdots \\
\alpha_{i_{2c}} &= (\phi_{i_{2c-1} i_{2c}} + \phi_{i_{2c} i_{2c-1}}) - \cdots + (\phi_{i_1 i_2} + \phi_{i_2 i_1}) - \alpha_{i_1} \\
\alpha_{i_1} &= (\phi_{i_{2c} i_1} + \phi_{i_1 i_{2c}}) - (\phi_{i_{2c-1} i_{2c}} + \phi_{i_{2c} i_{2c-1}}) + \cdots - (\phi_{i_1 i_2} + \phi_{i_2 i_1}) + \alpha_{i_1}.
\end{aligned}$$

From the last equation we derive that:

$$(\phi_{i_{2c} i_1} + \phi_{i_1 i_{2c}}) - (\phi_{i_{2c-1} i_{2c}} + \phi_{i_{2c} i_{2c-1}}) + \cdots - (\phi_{i_1 i_2} + \phi_{i_2 i_1}) = 2\pi m$$

for some $m \in \mathbb{Z}$. This completes the first implication of the theorem.

Let us focus on proving the second implication. Let $S = (\alpha, \delta)$ be output schedule of `SynchronizationAlgorithm`$(G, \{A, B\})$. From Corollary 2.2.21 we know there is a spanning tree T of G such that S synchronizes T and $V|S = \{A, B\}$. Let (i, j) be an edge of G which is not in T. Let $P = [i = v_1, \ldots, v_{2c} = j]$ be the

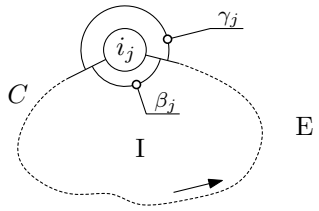


Figure 2.6: i_j is a vertex of a cycle C. Let "I" and "E" denote the interior- and exterior- region of C, respectively. Let β_j and γ_j denote the internal and external angle of C at vertex i_j, respectively.

path from i to j in T. Notice that P has an even number of vertices because G is bipartite. From hypothesis we know that:

$$(\phi_{v_1v_2}+\phi_{v_2v_1})-\cdots+(\phi_{v_{2c-1}v_{2c}}+\phi_{v_{2c}v_{2c-1}})-(\phi_{v_{2c}v_1}+\phi_{v_1v_{2c}})=2\pi m,\ m\in\mathbb{Z}. \quad (2.3)$$

From Lemma 2.2.9 we get that:

$$\begin{aligned}
\alpha_{v_2} &= (\phi_{v_1v_2}+\phi_{v_2v_1})-\alpha_{v_1}\\
\alpha_{v_3} &= (\phi_{v_2v_3}+\phi_{v_3v_2})-(\phi_{v_1v_2}+\phi_{v_2v_1})+\alpha_{v_1}\\
&\vdots \quad\quad \vdots\\
\alpha_{v_{2c}} &= (\phi_{v_{2c-1}v_{2c}}+\phi_{v_{2c}v_{2c-1}})-\cdots+(\phi_{v_1v_2}+\phi_{v_2v_1})-\alpha_{v_1}.
\end{aligned}$$

Then, using equation 2.3, we finally get that:

$$\begin{aligned}
\alpha_{v_{2c}} &= 2\pi m+(\phi_{v_{2c}v_1}+\phi_{v_1v_{2c}})-\alpha_{v_1}\\
&= (\phi_{v_{2c}v_1}+\phi_{v_1v_{2c}})-\alpha_{v_1}.
\end{aligned}$$

Therefore, $C_{v_{2c}}$ is synchronized with C_{v_1} (Lemma 2.2.9). This completes the proof. □

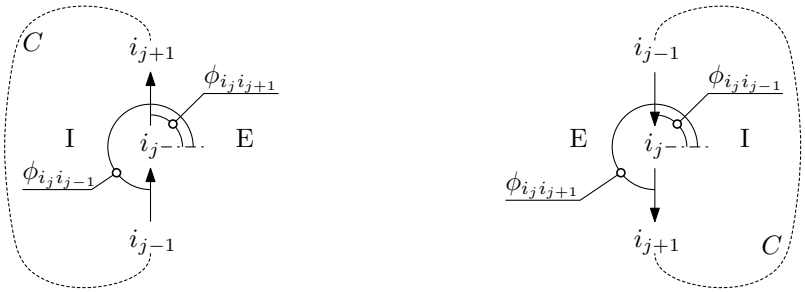
Let $C=\langle i_1,\ldots,i_{2c},i_1\rangle$ be a cycle of G with the vertices numbered in counter-clockwise ordering. Notice that traveling from i_j to i_{j+1} the interior of C is in the left side and the exterior of C is in the right side (see Figure 2.6). Let β_j and γ_j denote the internal and external angle of C at vertex i_j, respectively. Rewrite the expression of Theorem 2.2.30 grouping the link positions of a same trajectory:

$$(\phi_{i_1i_2}-\phi_{i_1i_{2c}})+(\phi_{i_2i_1}-\phi_{i_2i_3})+(\phi_{i_3i_4}-\phi_{i_3i_2})+\cdots+(\phi_{i_{2c}i_{2c-1}}-\phi_{i_{2c}i_1})\in 2\pi\mathbb{Z}. \quad (2.4)$$

For all $1\leq j\leq 2c$ notice that: if j is odd then the group is written as $(\phi_{i_ji_{j+1}}-\phi_{i_ji_{j-1}})$ and, if j is even then the group is written as $(\phi_{i_ji_{j-1}}-\phi_{i_ji_{j+1}})$.

Suppose that j is even. If $\phi_{i_ji_{j-1}}>\phi_{i_iji_{j+1}}$ then the value of the j-th group is:

$$(\phi_{i_ji_{j-1}}-\phi_{i_ji_{j+1}})=\beta_j,$$



(a) (b)

Figure 2.7: i_{j-1}, i_j and i_{j+1} are three consecutive vertices of a cycle visited in counterclockwise direction. "I" denotes the interior of the cycle and "E" denotes the exterior of the cycle. (a) $\phi_{i_j i_{j+1}} < \phi_{i_j i_{j-1}}$ and (b) $\phi_{i_j i_{j+1}} > \phi_{i_j i_{j-1}}$.

see Figure 2.6 and Figure 2.7a. If $\phi_{i_j i_{j-1}} < \phi_{i_i j i_{j+1}}$ then the value of the j-th group is:

$$(\phi_{i_j i_{j-1}} - \phi_{i_j i_{j+1}}) = -\gamma_j = \beta_j - 2\pi = \beta_j,$$

see Figure 2.6 and Figure 2.7b.

Suppose that j is odd. If $\phi_{i_j i_{j-1}} > \phi_{i_i j i_{j+1}}$ then the value of the j-th group is:

$$(\phi_{i_j i_{j+1}} - \phi_{i_j i_{j-1}}) = -\beta_j = \gamma_j - 2\pi = \gamma_j,$$

see Figure 2.6 and Figure 2.7a. If $\phi_{i_j i_{j-1}} < \phi_{i_i j i_{j+1}}$ then the value of the j-th group is:

$$(\phi_{i_j i_{j+1}} - \phi_{i_j i_{j-1}}) = \gamma_j,$$

see Figure 2.6 and Figure 2.7b. Doing the corresponding substitutions in equation 2.4 results:

$$\gamma_1 + \beta_2 + \cdots + \gamma_{2c-1} + \beta_{2c} \in 2\pi\mathbb{Z}.$$

Taking into account that $\beta_j + \gamma_j = 2\pi$ we get that:

$$\beta_1 + \gamma_2 + \cdots + \beta_{2c-1} + \gamma_{2c} \in 2\pi\mathbb{Z}.$$

Then, Theorem 2.2.30 can be rewritten as follows:

Theorem 2.2.31. *Let $G = (\{A, B\}, E)$ be a connected bipartite spanning graph of a system. G is synchronizable by $\{A, B\}$ if and only if every cycle $C = \langle i_1, \ldots, i_{2c}, i_1\rangle$ in G satisfies: $\beta_1 + \gamma_2 + \cdots + \beta_{2c-1} + \gamma_{2c} \in 2\pi\mathbb{Z}$ or equivalently: $\gamma_1 + \beta_2 + \cdots + \gamma_{2c-1} + \beta_{2c} \in 2\pi\mathbb{Z}$, where β_j and γ_j denote the internal and external angle of C at vertex i_j, respectively. Moreover, the output schedule S of* `SynchronizationAlgorithm`$(G, \{A, B\})$ *verifies that G is synchronizable by $\{A, B\}$.*

From this theorem the following result is directly derived:

Corollary 2.2.32. *Let T be a set of trajectories and let ϵ be the communication range of the robots. The synchronized system that ensures maximum number of synchronized pairs of robots where every pair of synchronized robots are moving in opposite directions is determined by the bipartite subgraph G of G_ϵ with maximum number of edges that fulfills: for every cycle $C = \langle i_1, \ldots, i_{2c}, i_1 \rangle$ in G, $\beta_1 + \gamma_2 + \cdots + \beta_{2c-1} + \gamma_{2c} \in 2\pi\mathbb{Z}$ or equivalently: $\gamma_1 + \beta_2 + \cdots + \gamma_{2c-1} + \beta_{2c} \in 2\pi\mathbb{Z}$, where β_j and γ_j denote the internal and external angle of C at vertex i_j, respectively.*

In this thesis the problem of computing the maximum subgraph G of G_ϵ such that G fulfills the conditions established in Corollary 2.2.32 remains open. However, in many practical applications the graph G is given (see the experiment description of Section 2.6). Nevertheless, Algorithm 3 shows an approach to get a synchronization schedule where every pair of synchronized robots travel in opposite directions.

Algorithm 3: Computing a synchronization schedule where all the synchronized pairs of robots are traveling in opposite directions.

```
   Input  : T, ε
   Output: α, δ
 1 G_ε ← computeIntersectionGraph(T, 1 + ε/2)
 2 if G_ε is bipartite then
 3 |   G ← G_ε
 4 else
 5 |   G ← maximumBipartiteSubgraph(G_ε)
 6 end
 7 A, B ← bipartition(V(G), G)
 8 G' ← maximumSynchronizableSubgraph(G)
 9 return SynchronizationAlgorithm(G', A)          // Algorithm 1
10
```

Th subrutine `bipartition` returns the bipartition of the set of vertices of a connected bipartite graph. Notice that this process takes linear time (recall that the number of edges of G s linear because G is a planar graph). The subrutine `maximumSynchronizableSubgraph`(G) implements some heuristic to find a big subgraph whose cycles fulfills the condition of Corollary 2.2.32. Notice, this subrutine is optional, using G in line 9 we get a synchronization schedule (Corollary 2.2.21) but, the obtained schedule may not maximize the number of synchronized pairs of robots.

2.3 Generalization to non-circular trajectories and heterogeneous robots

So far we have assumed homogeneous robots, with equal speeds and fuel capacities. In this section we describe how to extend the results under a more general and more realistic setting. Let us consider a team of n robots with different capabilities and performing their corresponding tasks through arbitrary disjoint closed trajectories $\{P_1, \ldots, P_n\}$.

Naturally, we have some control on the speeds of the robots as they can accelerate or decelerate to increase or decrease their speed. The speed can even change on different sections of the same trajectory. The key to generalizing the synchronization scheme used in the simple theoretical model is to force all members of the team to take (approximately) the same time to make a tour of their respective trajectories. In the ideal theoretical model two neighboring robots reach the communication link at the same time. In a real implementation of these strategies, however, we need to allow some margin of error in order to guarantee robustness. This can be accomplished by having each robot comparing its current and target locations at regular time intervals, and adjusting its speed (accelerating or decelerating), if necessary, in order to keep the synchronization. We refer to the time that a robot takes to make a tour as the *system period*.

We can use the communication links to partition a trajectory P_i into sections and assign each of them a travel time t_j such that $\sum_j t_j = \tau$, where τ is the system period. From this assignment we can compute the required speeds in each section. Once we know the required speed for each section, it suffices to know the starting position and movement direction of a robot in order to compute its position at any time. Thus, the problem in the general case is: given a partition of the trajectories into sections, assign a time to each section so that the cumulative time of all sections of a same trajectory is τ, and compute the initial position for every robot in its trajectory such that the system is synchronized.

The construction of the graph of potential links is similar to the case of the simplified model. We include a node in the graph for each trajectory P_i and include an edge between two nodes if the minimum distance between the respective trajectories is within the communication range. Note that we allow at most one edge between two nodes. If there are multiple pairs of points on two trajectories whose distances are within the communication range we choose the closest pair. Due to robustness reasons we are going to allow that the robots change its trajectories in some situations (see Section 2.4), then, to build the graph of potential links it is convenient to consider the communication range ϵ of the system as the minimum communication range between all the robots. In this way, every two robots can establish a communication link between any pair of neighboring trajectories. To compute the edges we can use known algorithms from computational geometry (e.g., [141, 105]) to efficiently find the minimum distance between two polygons.

In many cases, this step can be skipped because the communication links between the trajectories are established previously (see the experiment description of Section 2.6).

Then, let $T = \{P_1, P_2, \ldots, P_n\}$ be a set of disjoint closed trajectories assigned to a team of n robots. Also, let $G_\epsilon = (V, E_\epsilon)$ be the computed graph of potential links on T using the communication range $\epsilon = \min\{r_i\}$. Borrowing notation from previous sections, we use ϕ_{ij} to denote the location of P_i closest to P_j, for $(i, j) \in E_\epsilon$. (Note that in the generalization ϕ_{ij} is not an angle as in circular model, but rather a location in the curve P_i that can be parameterized to a value in $[0, 2\pi)$ if so desired). The following definitions generalize the main definition presented in the simple circular model.

Definition 2.3.1. *Let $T = \{P_1, \ldots, P_n\}$ be the set of trajectories of a system and let ϵ be the communication range. Let $G_\epsilon = (V, E_\epsilon)$ be the graph of potential links of the system. Let τ be the system period. A* schedule *of the system is a function $S : \mathbb{R}_{\geq 0} \to P_1 \times \cdots \times P_n$ such that $S = (s_1, \ldots, s_n)$ where $s_i : \mathbb{R}_{\geq 0} \to P_i$ is a function that indicates the expected position of a robot in P_i at any time instant, that is, $s_i(t)$ is the position of a robot in P_i at time t. Notice that $S(t) = S(t + k\tau)$ for all $t \in \mathbb{R}_{\geq 0}$ and $k \in \mathbb{N}$.*

Definition 2.3.2. *Let $T = \{P_1, \ldots, P_n\}$ be the set of trajectories of a system and let ϵ be the communication range. Let $G_\epsilon = (V, E_\epsilon)$ be the graph of potential links of the system. Let $S = (s_1, \ldots, s_n)$ be a schedule of the system. Two trajectories P_i and P_j are* synchronized *by S if $(i, j) \in E_\epsilon$ and $s_i(t) = \phi_{ij}$ then $s_j(t) = \phi_{ji}$. The* communication graph *of S is the spanning subgraph G of G_ϵ whose edges are the pairs (i, j) such that P_i and P_j are synchronized. For all subgraph H of G we say that H is* synchronizable *by S. If G is connected then S is a* synchronization schedule.

Let G be a connected spanning subgraph of G_ϵ to by synchronized. If G is a tree, then considering each trajectory as consisting of a single section, we obtain that the speed in each trajectory P_i is constant and equal to the ratio l_i/τ where l_i is the length of P_i. Then fixing arbitrary the initial position in trajectory P_i, we can compute the initial position of every neighbor of P_i as follows. Let P_j be a neighbor of P_i and suppose that a robot takes t units of time to reach ϕ_{ij} from its starting position. Thus P_i and P_j are synchronized if a robot takes t units of time to reach ϕ_{ji} from its starting position (see Figure 2.8). Note that the computed initial position for P_j depends on its chosen direction of movement. It is easy to see that in this way we can compute the initial positions for all the trajectories in the synchronized system. If G contains cycles, this simple approach does not work because considering each trajectory as a single section forces the robots to use constant speed (possibly different from each other) [2].

[2]However, it easy to see that the same approach that worked for unit circles would work here

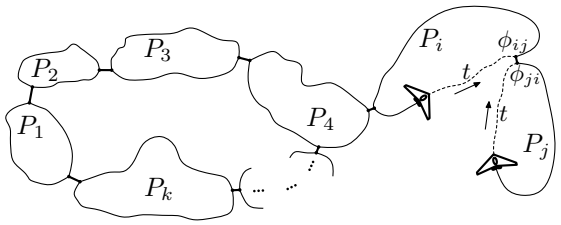


Figure 2.8: The dashed-dotted stroke is the section of trajectory to take to achieve ϕ_{ij} and ϕ_{ji} respectively. Both should use the same amount of time (t) to achieve the link position.

Figure 2.8 shows a cycle in the graph G, if we know the starting position in P_1 we can compute the starting position in P_2, then in P_3, and so on until we have computed the starting position in P_k using the starting position in P_{k-1}. But having done this, P_1 and P_k are not necessarily synchronized. In the following we are going to extend the results of sections 2.2.2 and 2.2.3 to help us determine when a graph can be synchronized with robots moving in the same direction or with neighbors moving in opposite directions, respectively.

First, we introduce notation needed for the rest of this section. Let β_i (resp. γ_i) be the time it takes a robot to travel the inside (resp. outside) section of trajectory P_i in the cycle (see Figure 2.9). We use elapsed time to describe the location of robots as follows: $\phi_{ij}+t$ denotes the position in P_i obtained by moving CCW from ϕ_{ij} during t units of time. Analogously, $\phi_{ij}-t$ denotes the position obtained by moving CW in P_i from ϕ_{ij} during t units of time.

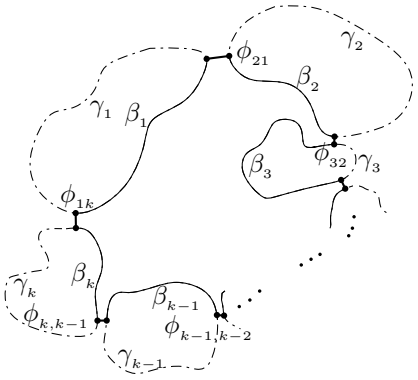


Figure 2.9: A cycle in a synchronized system.

Theorem 2.3.3. *Let τ be the period of a system whose graph of potential links is G_ϵ. Let G be a connected spanning subgraph of G_ϵ. G is synchronizable by a*

if we parameterize position to a value in $[0, 2\pi)$ and allow the robots to change its speed to cover different distances in physical space but equal distance in parameter space.

schedule where all the robots move in the same direction if and only if for all cycle $C = \langle P_1, P_2, \ldots, P_k \rangle$ *in* G *there exists a value* $z \in \mathbb{N}$ *such that:*

$$\beta_1 + \beta_2 + \cdots + \beta_k = z\tau \quad \textit{and}$$
$$\gamma_1 + \gamma_2 + \cdots + \gamma_k = (k - z)\tau,$$

where β_i *and* γ_i *are the required times to travel the internal section and external section of* C *at trajectory* P_i*.*

Proof. In the equations below we use α_i to denote the starting position in P_i. Without loss of generality, suppose that the robots move CCW and that $\alpha_1 = \phi_{1k}$. Computing $\alpha_2, \alpha_3, \ldots, \alpha_k$ we obtain the following result:

$$\begin{aligned}
\alpha_1 &= \phi_{1k} \\
\alpha_2 &= \phi_{21} - \beta_1 \\
\alpha_3 &= \phi_{32} - \beta_1 - \beta_2 \\
&\vdots \qquad\qquad \vdots \\
\alpha_k &= \phi_{k,k-1} - \beta_1 - \beta_2 - \cdots - \beta_{k-1}
\end{aligned}$$

Obviously, having that P_i is synchronized with P_{i+1} for $1 \leq i < k$ then the cycle can be synchronized if and only if $\alpha_1 = \phi_{1k} - \beta_1 - \beta_2 - \cdots - \beta_k = \phi_{1k}$, from here we deduce that $\beta_1 + \beta_2 + \cdots + \beta_k$ is a multiple of τ because $\phi_{ij} = \phi_{ij} + \tau$. Note that $\beta_i < \tau$ for all $1 \leq i \leq k$, then

$$\beta_1 + \beta_2 + \cdots + \beta_k = z\tau \quad z \in \mathbb{N}, (0 < z < k)$$

Since $\beta_i + \gamma_i = \tau$ then

$$(\beta_1 + \gamma_1) + (\beta_2 + \gamma_2) + \cdots + (\beta_k + \gamma_k) = k\tau$$

The difference between (2.3) and (2.3) is

$$\gamma_1 + \gamma_2 + \cdots + \gamma_k = (k - z)\tau.$$

□

Note that k is not necessarily an even number, but in the simplified model $\beta_1 + \beta_2 + \cdots + \beta_k$ is a multiple of τ only if k is even. This is consistent with the bipartition requirement.

Now, consider the case of neighbors moving in opposite directions. In this case, G must be bipartite.

Theorem 2.3.4. *Let* τ *be the period of a system whose graph of potential links is* G_ϵ*. Let* G *be a connected spanning bipartite subgraph of* G_ϵ*.* G *is synchronizable*

by a schedule where every pair synchronized robot move in opposite directions if and only if for all cycle $C = \langle P_1, P_2, \ldots, P_{2k}\rangle$ in G there exists a value $z \in \mathbb{N}$ such that:

$$\beta_1 + \gamma_2 + \beta_3 + \cdots + \beta_{2k-1} + \gamma_{2k} = z\tau \quad \textit{and}$$
$$\gamma_1 + \beta_2 + \gamma_3 + \cdots + \gamma_{2k-1} + \beta_{2k} = (2k - z)\tau,$$

where β_i and γ_i are the required times to travel the internal section and external section of C at trajectory P_i.

Proof. In the equations we use α_i to denote the starting position of in P_i. Without loss of generality, suppose that $\alpha_1 = \phi_{1k}$ and P_1 has CCW direction. Computing $\alpha_2, \alpha_3, \ldots, \alpha_k$ we obtain the following result:

$$\begin{aligned}
\alpha_1 &= \phi_{1k} \\
\alpha_2 &= \phi_{21} + \beta_1 \\
\alpha_3 &= \phi_{32} - \beta_1 - \gamma_2 \\
\vdots &\qquad\quad \vdots \\
\alpha_{2k-1} &= \phi_{2k-1,2k-2} - \beta_1 - \gamma_2 - \cdots - \gamma_{2k-2} \\
\alpha_{2k} &= \phi_{2k,2k-1} + \beta_1 + \gamma_2 + \cdots + \gamma_{2k-2} + \beta_{2k-1}
\end{aligned}$$

Obviously, having that P_i is synchronized with P_{i+1} for $1 \leq i < 2k$ implies that the cycle can be synchronized if and only if $\alpha_1 = \phi_{1k} - \beta_1 - \gamma_2 - \cdots - \beta_{2k-1} - \gamma_{2k} = \phi_{1k}$, from here we conclude that $\beta_1 + \gamma_2 + \cdots + \beta_{2k-1} + \gamma_{2k}$ is a multiple of τ because $\phi_{ij} = \phi_{ij} + \tau$. Note that $\beta_i < \tau$ for all $1 \leq i \leq 2k$, then

$$\beta_1 + \gamma_2 + \cdots + \beta_{2k-1} + \gamma_{2k} = z\tau \quad z \in \mathbb{N}, (0 < z < 2k)$$

Since $\beta_i + \gamma_i = \tau$ then

$$(\beta_1 + \gamma_1) + (\beta_2 + \gamma_2) + \cdots + (\beta_{2k} + \gamma_{2k}) = 2k\tau$$

The difference between (2.3) and (2.3) is

$$\gamma_1 + \beta_2 + \cdots + \gamma_{2k-1} + \beta_{2k} = (2k - z)\tau.$$

□

2.4 Robustness

We now address the issue of robustness of the synchronized system. Imagine that one member u of the team becomes unavailable because of failure or because it needs to leave the system temporarily (e.g., to refuel). First, let us consider the

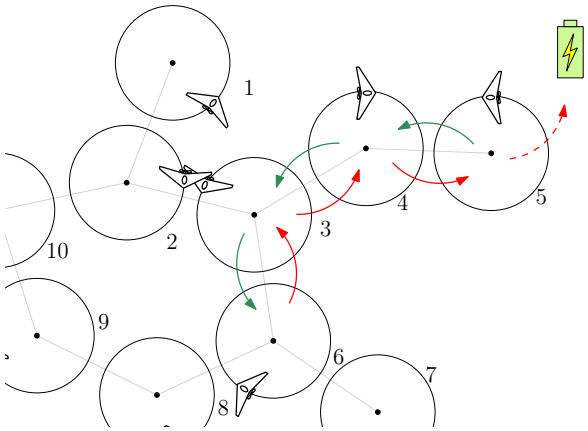


Figure 2.10: Robot 6 can leave the system swapping with 3, 4 and 5 in this order.

latter scenario. We want to minimize the detrimental effect of the departing robot u on global system performance. A simple strategy consists of repeatedly "swapping with a neighbor" until u reaches the trajectory nearest to the outside goal, at wich this point u can easily leave the system as illustrated in Figure 2.10. The best moment to make a swap with a neighbor is when the involved robots arrive to a link position (this is, indeed, another advantage of a synchronized system) where a simple collision avoidance approach between two cooperative vehicles can be applied. Consider now the case of a hard failure, where one (or more) robots fail and can no longer move. Under the assumption that robots which fail do not block live robots, we can easily solve the problem while maintaining the set of trajectories. In this case, one or more live neighbors can assume the tasks of the inoperative robots by using the same routes. If a robot arrives at a communication position and the neighbor is not there, then it assumes that this neighbor failed and passes to the neighboring trajectory performing the a task of the inoperative robot. If needed, even a new robot might be inserted in the system by using the swapping strategy mentioned above.

We call *shifting* to the process in which a robot passes from its current trajectory to a neighboring one. Notice that a swap consists of two shifting operations. When a robot passes to a neighboring trajectory it follows the movement direction assigned to this trajectory, see Figure 2.11. This behavior allows keeping the synchronization in the system even when the robots shift their trajectories. Notice that, during a shifting operation the robots must accelerate in order to adjust its time-position with the schedule in the new trajectory. Let ω be the minimum travel time between two consecutive link positions of a same trajectory in the system. Assume that if a robot in trajectory P_i at time t starts a shifting operation toward trajectory P_j then after ω time units it will be at the scheduled position of P_j, that is:

Property 2.4.1. *Let $T = \{P_1, \ldots, P_n\}$ be the set of trajectories of a synchronized*

system with synchronization schedule $S = (s_1, \ldots, s_n)$. Let ω be the minimum travel time between two consecutive link positions of a same trajectory in the system. Let P_i and P_j be two trajectories such that (i, j) is in the communication graph of S. If at time t a robot u in P_i (the position of u at time t is $s_i(t)$) starts a shifting operation toward P_j then, during this process, u accelerates if it is needed in order to reach the position $s_j(t + \omega)$ at time $t + \omega$ after that u continues with the programmed speed following the schedule in trajectory P_j.

In such a way a robot u, after a shifting operation from P_i to P_j, can reach the next link position in P_j at the scheduled time.

The following result establishes a non-redundancy property of robots occupying the trajectory of a inoperative neighbor.

Theorem 2.4.2. *In a synchronized system that fulfills Property 2.4.1 each trajectory is occupied by at most one robot at any time.*

Proof. If all the trajectories have at most one robot, then after a swapping operation all the trajectories will have at most one robot by definition. We focus now in the case when a robot shifts its trajectory because it does not detect the expected neighbor. Assume, for the sake of contradiction, that there exists a trajectory P_i occupied by two robots u and u' which entered P_i coming from the link positions ϕ_{ji} and ϕ_{ki} in P_j and P_k, respectively. Robots u and u' could not have entered P_i at the same time because a robot in P_i cannot be at ϕ_{ij} and ϕ_{ik} simultaneously. Suppose that u' entered second. When u' arrives at ϕ_{ki}, u is already moving in P_i and has reached ϕ_{ik} (by Property 2.4.1). As a consequence, robot u' does not enter in P_i because it detects that P_i is occupied by robot u, a contradiction. □

See Figure 2.11, notice that a shifting operation between two trajectories with opposite directions is smoother than a shifting operation between trajectories with the same direction. So, for a mobile robot is easier to perform a shifting operation between trajectories with opposite travel directions and even more taking into account that the robot must accelerate during this operation. Moreover, if we are using aerial robots in the system the kinematic constraints of these vehicles prevent the passing from one trajectory to another keeping the initial assigned movement direction if the trajectories have the same direction. Therefore, the opposite movement directions in the neighboring trajectories ensures the robustness of the approach.

Recall that two robots don't need to be exactly a the corresponding link positions at the same time to establish a communication link, there is a communication region around the link positions where the robots can exchange information, see Figure 2.2. Notice that some external elements may affect the precision of the robots motion following the schedule (e.g., aerial robots performance may be affected by the lack of accuracy of the location sensors or the wind), then it could happen that a robot arrives at a link position with some delay or advance. The

(a) (b)

Figure 2.11: Path (bold stroke) traveled by a robot (hollow point) when it passes from a trajectory to another. (a) Both trajectories have the same direction. (b) The trajectories have opposite directions.

communication regions allows the system to *absorb* these perturbations in the planned schedule. Notice that opposite travel directions between synchronized neighbors guaranties larger communication regions, see Figure 2.2. The larger the communication regions, the greater the delays or advances that the system can tolerate. Therefore, using a schedule where all the synchronized pairs of robots move in opposite directions assures a better performance of the system.

In the rest of this thesis we consider synchronized systems where every pair of synchronized trajectories have assigned opposite travel directions and the robots perform shifting operations to swap their trajectories (when it is required) or when they detect the absence of a neighboring robot in a link position.

Notice that, to deploy such system, every robot requires all the information regarding to the set of trajectories and the link positions between them. They also require the synchronization schedule function of the system. When the system is deployed, every robot moves toward initially assigned trajectory at the starting position and loads a motion plan to follow the schedule in the current trajectory. When the mission starts every robot repeatedly executes the pseudo-code shown in Algorithm 4. The first line of the algorithm adjusts the velocity of the robot in order to maintain a tracking of the scheduled position. Notice that P' is the neighboring trajectory and P is the current trajectory, line 5 completes the shifting process because the neighboring trajectory becomes the current one. The function `load_trajectory_plan`$(P,$ `current_pos`$,$ `current_time`$)$ load a motion plan to follow the schedule on trajectory P from the current position and the current time. And the function `load_shift_plan`(P, P') compute and load a motion plan to pass from P to P'. A video is available at `https://www.youtube.com/watch?v=T0V6tO8OHOI` illustrating all the phases of the algorithm and the motion of the robots when the system is deployed.

2.5 Simulation and computational results for area surveillance with aerial robots

Consider the cooperative surveillance of an area by means of a team of small fixed wing aerial robots (ARs). The area can be divided into a grid whose cells have a

Algorithm 4: Onboard algorithm

```
adjust_velocity()
if shifting then
    if current_pos is in P' then
        shifting ← false
        P ← P'
        load_trajectory_plan(P, current_pos, current_time)
    end
else
    do_work()
    if entering in a communication region then
        open_communication()
        P' ← neighboring_trajectory()
    end
    if communications are opened then
        if sharing_info then
            if end of info stream then
                sharing_info ← false
                close_communication()
                if a swapping is required then
                    shifting ← true
                    load_shift_plan(P, P')
                end
            end
        if a neighbor has been detected then
            start_info_exchange()
            sharing_info ← true
        if reaching the end of the communication region then
            close_communication()
            shifting ← true
            load_shift_plan(P, P')
        end
    end
end
```

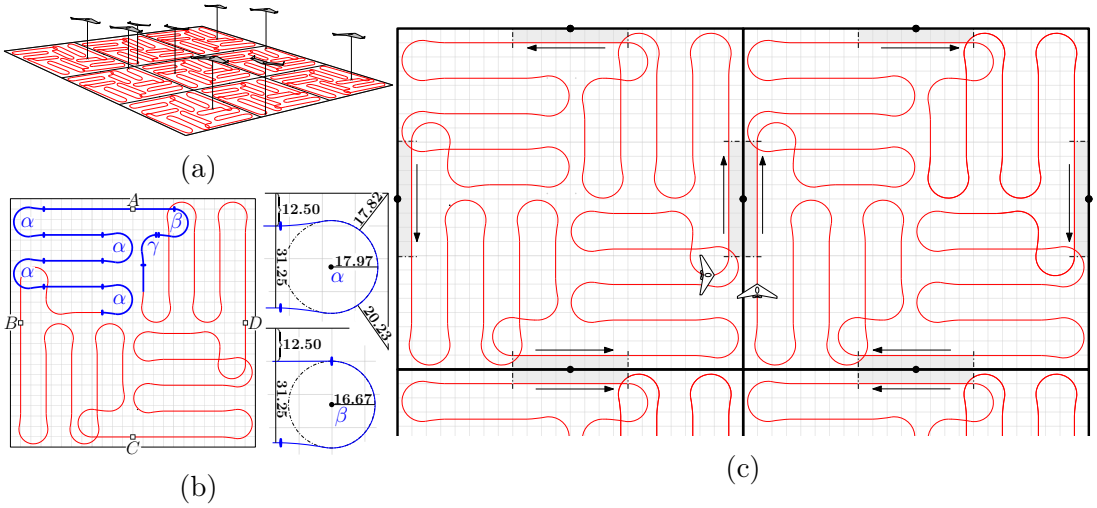


Figure 2.12: (a) A team of fixed wing ARs is surveying a 3×3 grid divided region whose cells are 300m×300m. Each AR is flying in its cell over a predefined trajectory. (b) The trajectory in a cell of the grid is formed by four repetitions of the pattern drawn with thick stroke. The turns has been noted with the Greek letters α, β and γ, and using the same letter for equal turns. The turn γ is the half of turn α. On the right, the measures of the turns α and β. (c) Section of a grid division. The communication region between them is shaded in gray. The aerial robots have been magnified to make them look better.

size of 300m×300m and each cell is assigned to a member of the team. Figure 2.12a shows a rectangular region divided into a grid of three rows by three columns (in short, a 3×3 grid). The robots are equipped with an on-board camera with focal length 14mm and a field of view of $53.13° \times 36.87°$. The robots are programmed to fly at a constant altitude of 90m (to meet regulations preventing the obstruction of commercial air traffic) and at a constant speed of 12m/s. From this altitude the covered area by the camera is approximately 45m×30m and targets greater than 0.06m can be detected. To cover a 300m×300m cell we can use a *back-and-forth* closed trajectory (see for example [147]) as it is shown in Figure 2.12b. Note that the farthest points to the trajectory are at a distance of 20.23m, so in these critical points we have a deviation margin on the route of ±2m. Thus, every point can be watched by the camera in some instant during the tour. The kinematic constraints of the fixed-wing aircraft imposes a turning radius greater than 16.67m at 12m/s as shown in Figure 2.12b.

The robots have very constrained communication range and the only possibility to exchange relevant information is to synchronize the robots in such a way that two robots meet in adjacent cells. The objective is to synchronize the system such that all the robots transmit the collected information in spite of robot failures and dealing with required refueling maneuvers as it was mentioned in Section 2.1. In

Table 2.1: Simulations results surveying grid divided regions.

	N° ARs	**N° F. ARs**	**Avg. BT(s)**	**Max. IT(s)**
Grid 3×3	9	0	348.71	0.00
Grid 3×3	9	2	394.46	157.87
Grid 3×3	9	4	482.33	277.50
Grid 5×3	15	0	543.30	0.00
Grid 5×3	15	3	595.50	260.63
Grid 5×3	15	7	658.43	607.50

our scenario the link position between two adjacent trajectories is at the middle point of the common side between the corresponding cells, see Figure 2.12. The distance between two neighboring trajectories at the communication link is 25m (see Figure 2.12c), which is close enough to share information using the on-board wireless equipment and sufficiently far to avoid collisions. The proposed trajectory is symmetrical and formed by the union of four repetitions of the pattern shown in Figure 2.12b with thick stroke. This symmetry implies that an aerial robot makes the same route traveling from any link position to the next link position(Figure 2.12b shows that the trajectory segments AB, BC, CD and DA are equal). Note that all the cycles that appear in the graph of potential links induced by the grid division are equal and formed by four trajectories. Theorem 2.3.4 ensures that the graph of potential links is synchronizable on the described situation.

Within this scheme, any two synchronized neighbors are moving in opposite directions, one of them in CW direction and the other in CCW direction, although they move along the same direction in the communication region (see Figure 2.12c). The communication region between two neighboring robots has a length of 100m. Thus, flying at 12m/s, they have 8.33s to share information. The length of the proposed covering trajectories is approximately 3609.79m and an AR spends 300.82s to make a tour in its cell.

Simulations were performed to validate the proposed approach. We ran each simulation during 50 periods, 15 041s. We considered two measures to evaluate the performance of the system: the *broadcast time (BT)*, that is, the time it takes for a message issued by a robot to reach the full team, and the *idle time (IT)*, defined as the interval of time in which a trajectory is not attended by any AR.

Table 2.1 shows the results of the experiments on 3×3 and 5×3 grid graphs. The first column shows the initial number of ARs; the second column, the number of ARs with critical failures (selected randomly); the third, the broadcast time; and the last one shows the maximum idle times during the simulation. All the results correspond to the average of 10 simulations with the same parameters.

In addition, we performed other simulations using communication graphs ran-

Table 2.2: Simulations results using graphs randomly generated.

	N° ARs	**N° F. ARs**	**Avg. BT(s)**	**Max. IT(s)**
Rand10	10	0	209.69	0.0
Rand10	10	2	216.66	87.2
Rand10	10	5	273.92	207.4
Rand15	15	0	237.50	0.0
Rand15	15	3	293.11	76.2
Rand15	15	7	348.28	231.9

domly generated and circular trajectories. A random graph is generated constructively as follows. Suppose that we have a connected graph formed by m circles. In order to increase the graph we select a random circle c and a random ray r from the center of c, then the new circle is placed with center on r such that it is disjoint with the other circles and keeping the connection in the graph. With this approach (starting with one circle) we can construct graphs of any numbers of nodes. Table 2.2 (with the same structure of Table 2.1) shows the results of the simulations with random graphs of 10 and 15 trajectories. Circles of radius 150m and a system period of 80s were considered and, therefore, the ARs are flying at 11.78m/s. Each simulation was ran for 4000s.

Tables 2.1 and 2.2 show that our method is robust. The broadcast time does not grow significantly even if near 50% of the robots fail. Thus, the system is fault-tolerant. Also, it is worth noting the importance of a large number of communication links in the graph. In grid configurations, all the cycles meet the hypothesis of Theorem 2.3.4 and we can use the graph of potential links is synchronizable. However, in random graphs the probability to generate cycles fulfilling such hypothesis is low. In this case, the algorithm computes the feasible maximal subgraph losing many communication links. For random graphs, the idle time measure shows a decrease in performance. In the next chapter of this thesis we focus on the study of the effect of leaving robots.

2.6 Preliminary experimental results

In order to have some additional hints on the applicability of the synchronization strategies previously described, some preliminary flights were performed on the indoor multi-UAV testbed of the Center for Advanced Aerospace Technologies (CATEC) located in Seville (Spain). This testbed has been used in the last years to develop and test cooperation algorithms for multiple aerial vehicles. The useful volume where tests can be conducted is a box with a base of 14×14 meters and 5

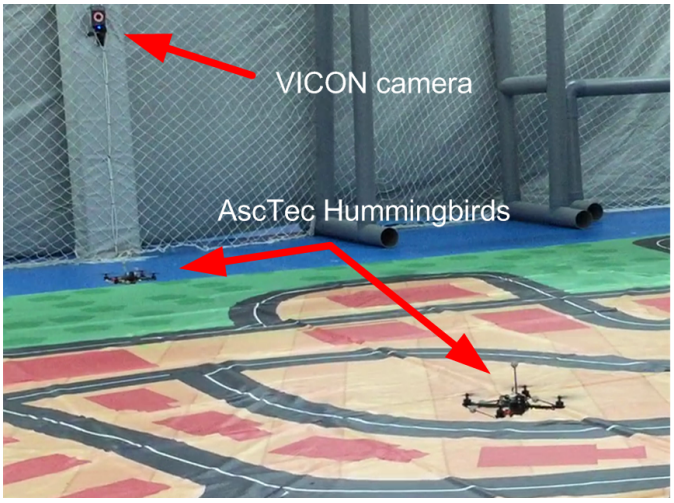


Figure 2.13: Photograph of the indoor testbed used for the preliminary experiments and the aerial vehicles involved.

meters height. The testbed has an indoor localization system based on 20 VICON cameras that only needs the installation of passive markers on each of the aerial robots. This system is able to provide, in real-time, the position and attitude of up to ten each aerial vehicles with centimeter accuracy. The quadrotors used in our experiments were two Hummingbirds from Ascending Technologies [3] which have a payload of 200 grams with a flight autonomy up to 20 minutes. Figure 2.13 shows the indoor testbed with a drawing on the floor that emulates the aerial view of an urban area with roads and buildings.

As two quadrotors are involved in the experiments, the indoor testbed arena is divide into two cells of 10×5 meters each. Each quadrotor is equipped with an on-board camera pointing downwards with a field of view of $85° \times 45°$. The flight altitude is fixed at 1.4 meters in such a way that the trajectories required to cover the whole urban setting are not as simple as two segments inside each sub-area. For this altitude, the trajectories computed as reference for the vehicles in order to cover the sub-areas are shown in Figure 2.14. The right and left trajectories are symmetrical and the communication region between the two trajectories is shadowed. The shifting paths drawn in dashed stroke are used if one of the aerial robots should be covering the whole area in case the other is lost. The communication segments are 3 meters long, and as a consequence of the opposite motion of the robots in their trajectories, in the communication region they fly in the same direction with zero relative speed between them. The distance between points B and C is 2 meters, and the length of the shifting path from A to B is approximately 2.50 meters long. In order to guarantee the synchronization between the robots, the time required to travel from C to B must be approximately equal to the time

[3] http://www.asctec.de/en/

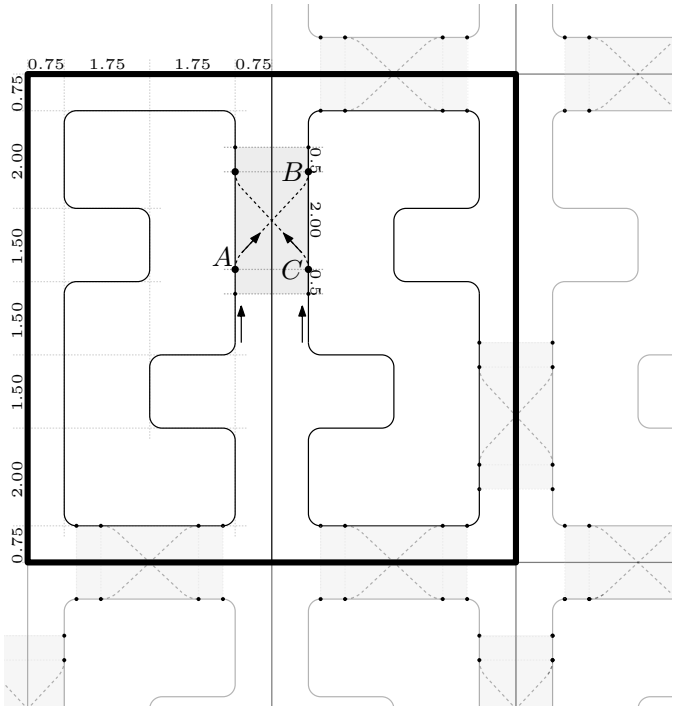


Figure 2.14: The square in bold is a 10×10-meter experimental area. This area is divided into two 10×5 sub-areas. The trajectories (with lengths in meters) used to cover the experimental area are shown. Also, the outside the square in bold suggests how to extend the pattern to greater areas by using more aerial robots.

required to travel from A to B. Therefore the speed in the shifting paths must be 1.25 times the speed from C to B.

At the beginning of the mission, both aerial robots are following their coverage paths exchanging information along the gray zone depicted in Fig. 2.14. Figure 2.15 shows the distance between the robots during the first two minutes until robot B leaves the arena. Once robot A does not meet robot B along the gray region, it extends its initial path to cover both sub-areas. Figure 2.16 shows the 3D trajectory followed by robot A during the whole mission. It can be seen how it follows the reference and extends its coverage area passing through the gray region until robot B comes back. The video of the whole mission can be downloaded from `https://grvc.us.es/TROsynch`.

This scenario can be extended to more aerial robots and greater areas as it is illustrated in Figure 2.14, where every pair of neighbors have identical communication region with identical shifting paths.

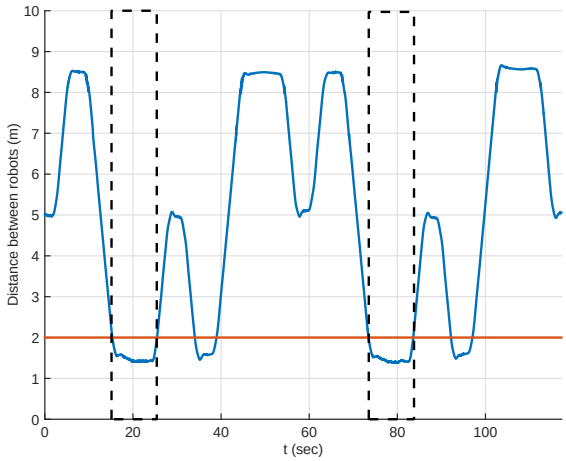


Figure 2.15: Distance between the aerial robots before robot B leaves the arena. It can be seen that the distance is below the threshold of 2 meters considered for the communication between the robots during the periods when they meet along the gray region depicted in Fig. 2.14. These periods are highlighted with dashed line rectangles.

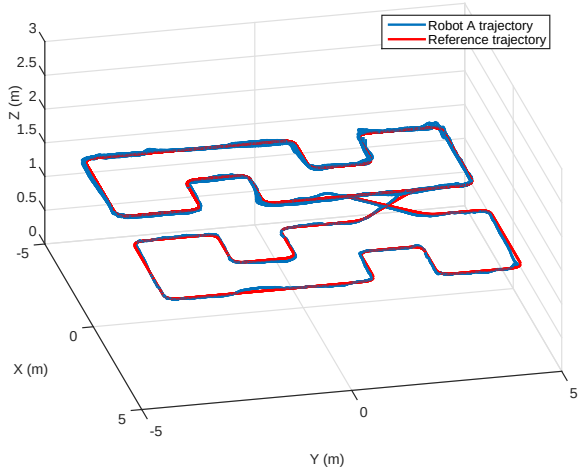


Figure 2.16: The trajectory in 3D followed by robot A during the whole mission is depicted in blue, whereas the reference is represented in red. Although there are some tracking errors, the size of the gray region allows to compensate the deviations.

2.7 Conclusions

In this chapter we have introduced a new problem on communication systems: Consider a set of n pairwise closed and simple trajectories and a set of n robots (one per trajectory) with a limited communication range ϵ. In this setting we say that there is a *communication link* between two trajectories if the minimum distance between them is less than ϵ. We consider, at most, a communication link for every pair of trajectories. We say that two robots moving along their trajectories are *synchronized* if they periodically arrive at the same time at the same communication link. Thus, a pair of synchronized robots can establishes a wireless communication and exchange information between them, periodically. The *synchronization problem* asks for a motion schedule for the system such that the number of synchronized pairs of robots is maximized.

We have proven that the optimum motion schedule is not unique, rather there is a whole family (equivalence class) of motion schedules that maximize the number of synchronized pairs of robots. We introduced the concept of *communication graph* as a graph whose vertices are the trajectories and edges are the pairs of trajectories with synchronized robots. Therefore, the synchronization problem looks for the schedule with larger communication graph. In a simple case where the trajectories are unit circles and the robots move at constant speed, we have shown that a family of schedules is determined by a communication graph and a bipartition $\{A, B\}$ of the trajectories set. This partition indicates that the travel direction assigned to robots in A's trajectories is opposite to the travel directions of robots in B's trajectories (i.e., if the robots in A's trajectories move in clockwise direction then the robots in B's trajectories move in counter-clockwise direction and, viceversa).

We have also studied the synchronization problem in two different specific configurations:

1. all the robots move in the same direction and,

2. all pairs of synchronized robots move in opposite directions.

In both cases the optimum family of schedules is determined only by the communication graph because the bipartition is directly derived from these settings: in the first case (1), the bipartition is $\{V, \emptyset\}$ (where V is the whole set of trajectories) and, in the second case (2), the communication graph G must be bipartite and the bipartition is formed by the partite sets of G. We have studied and characterized the optimum schedule in both cases. And, using the simple model where the trajectories are unit circles, we have shown how to compute a schedule that ensures that messages can be exchange (maybe passing through many robots) between any pair of robots. Also, we exposed an algorithm to solve the synchronization problem in the first case but, the second case remains as an open problem. Anyway, in many practical applications the graph G of communications is a priori established,

so, in these cases, to build a specific schedule with communication graph G we can compute the starting positions of the robots using our characterization.

For the sake of robustness, we described a recovery protocol, that maintaining the same set of trajectories and the initial synchronization scheme, reduce the detrimental effect of robot failures. Due to kinematic constraints in practical applications, it is convenient to apply this protocol on systems where all pairs of synchronized robots move in opposite directions. Our simulations and preliminary experiments show that the presented approach is robust against robot failures.

Future work could explore other practical variations of the problem, including: 1) the trajectories can overlap and share multiple communication links that provide extra opportunities for information exchange and 2) instead of failure, we can consider the inability of an agent to properly maintain its schedule along its trajectory.

Chapter 3

Resilience of a Synchronized Communication System

ROBUSTNESS is an important issue in distributed systems. Many research works are focused in the study of measures to determine how fault-tolerant a system is [54, 71, 145, 144]. The developing of these kind of metrics allows to compare different systems and determine which one is better. This kind of study also leads to determine what are the critical scenarios for some specific system.

In this chapter we are going to study more deeply the robustness of a synchronized system. We are going to base our study in the simplified model introduced in Section 2.2 of Chapter 2. The results that we present in this model can be extended to general synchronized system by using the ideas presented in Section 2.3 of Chapter 2.

In the simplified model all the trajectories are unit-circles, the communication range of the system is a value $\epsilon < 0.5$, all the robots move with constant speed along their trajectories and the system period (i.e., the required time by robot to make a complete tour in a trajectory) is one unit of time. Figure 3.1 shows a synchronized system in the simplified model. Recall that in a such systems, every pair of synchronized robots are moving in opposite directions (one clockwise and the other counterclockwise). We show in light gray the pairs of neighboring robots with an established communication link between them. Each subfigure represents the state of the system at every quarter of time unit, starting with the state shown in Figure 3.1a.

In a synchronized system, a trajectory swap between two neighboring robots is

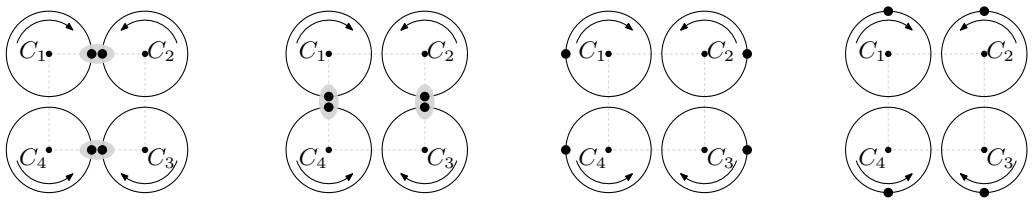


Figure 3.1: Synchronized system of robots following circular trajectories. The robots are represented as solid points in the circles. Arrows represent the movement direction of the robots in the trajectories. In this example the graph of potential links is a cycle of four nodes.

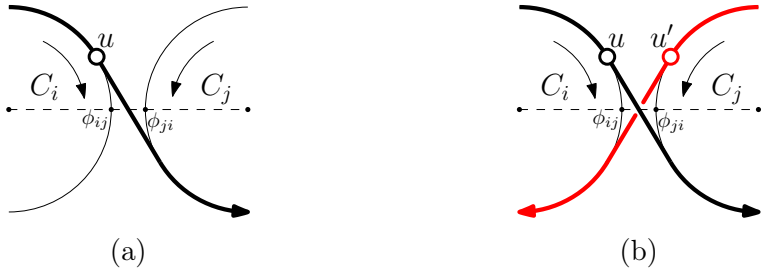
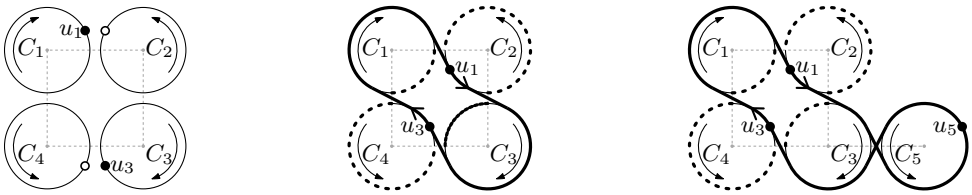


Figure 3.2: (a) A robot shifts to a neighboring trajectory when it detects that the corresponding neighboring robot has left the system. The robot u follows the path drawn with bold solid stroke. There were no robots on the trajectory C_j when u is arriving at the link position in C_i. (b) A swapping operation between robots u and u'.

allowed in some situations (see Section 2.4 of Chapter 2) and does not affect the performance of the system. See Figure 3.2b, notice that a swap operation takes place in a communication link, so, the involved robots can exchange information and after the swapping u' (resp. u) continues with the schedule in C_i (resp. C_j) performing the task that was previously doing u (resp. u'). Therefore, from a global point of view after a swapping the status of the system has not changed.

Now, let us see how the system is affected when one or more robots leave. The first thing to notice is: if k robots have left the system then there is always k unattended trajectories. In Section 2.4 of Chapter 2, a strategy was proposed to address this problem: when a robot u in C_i arrives to the link position ϕ_{ij} and detects no other robot in the neighboring trajectory C_j, it considers that C_j is unattended and shifts from C_i to C_j in order to assume the task in C_j, see Figure 3.2a. Notice that after the shifting operation C_i is unattended and C_j is not. Also notice that u has lost a chance of exchanging information. Section 2.5 of Chapter 2 presents some computational experiments showing that after randomly removing a few robots of the system, the surviving ones (non-removed robots) eventually meet



(a) (b) (c)

Figure 3.3: (a) and (b) When the robots represented by hollow points leave the system then the surviving ones, solid points, follow the closed path drawn with bold solid stroke. The trajectory segments in dotted stroke in (b) and (c) are non-covered. In (b) u_1 and u_3 are isolated, however, in (c), u_1 and u_3 are not isolated, they meet u_5 periodically. The message broadcasting is possible in (c) but it is not in (b).

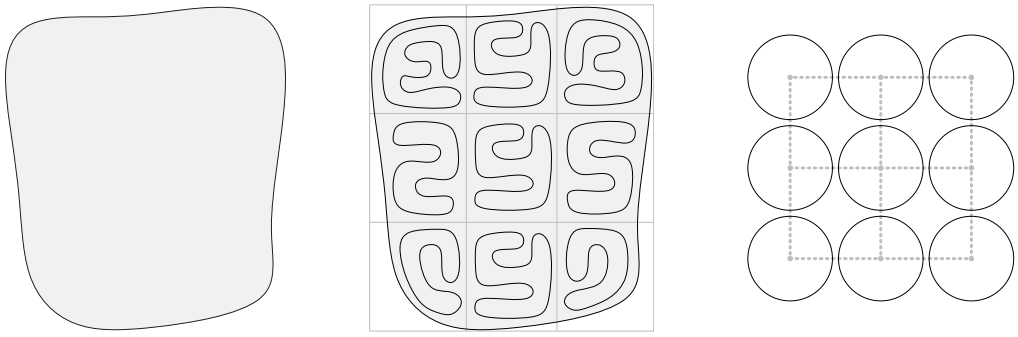
each others and visit all the trajectories. So, no trajectory is unattended forever and the communication remains (it is possible to make a broadcast between the surviving robots).

However, in some cases, if enough robots leave the team, an undesirable phenomenon may occur: a robot, independent of how much longer stays in the system, it permanently fails to encounter other robots every time it arrives at a link, causing it to repeatedly shift to neighboring trajectories. In this case, we say that the robot is *isolated* or in *isolation* mode. Figures 3.3 (a) and (b) show a synchronized system where two robots leave and the remaining robots, u_1 and u_3, permanently fail to encounter other robots at the link positions, so they enter the isolation state[1].

Another problem, more general than the isolation of some robots, is the loss of the ability for messages broadcasting in the system. It is easy to see that, even without isolated robots, the system may not allow broadcasting, as the surviving robots are partitioned into independent connected components, for communication purposes. We say that there is a *loss of connectivity* in a system if only one robot survives or there is a pair of robots that cannot exchange messages through a sequence of message exchanges between neighboring robots. For example, in Figure 3.3b, a broadcast is not possible because the living robots are isolated.

Now, focus on covering. Notice that in a system with one robot per trajectory, every point of every trajectory is visited by some robot periodically. We say that every trajectory point is *covered*. If some robots leave the system and the remaining ones stay in their trajectories (ignoring the shifting strategy altogether) then, obviously all the points of the trajectories of the leaving robots are *non-covered*. Using the shifting strategy one may think that the covering is guaranteed. However, this is not true. Sometimes, the departure of a set robots (independent of

[1] An illustration of this phenomenon is at `https://www.youtube.com/watch?v=64gKnefnXew`.



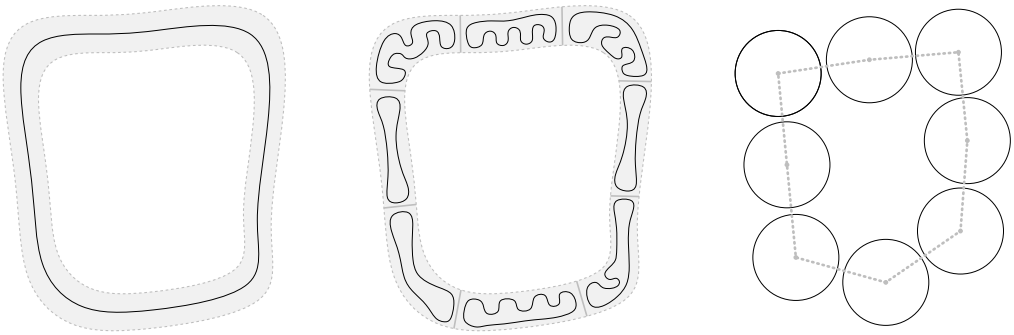
(a) (b) (c)

Figure 3.4: (a) Region to cover. (b) Area decomposition of the region using a grid pattern and the paths to cover every cell of the region. (c) Simplification of the practical model using an abstraction with circular trajectories. The underlying communication graph is represented using gray dotted strokes.

whether it causes a loss of communication or not among the active robots) results in some trajectory segments (set of consecutive points of a trajectory), or even entire trajectories, to no longer be visited by an active robot. In this case we say that these are *non-covered* trajectory segments, see Figure 3.3b and 3.3c.

In this chapters we are going to study these undesirable phenomenons and we are going to establish resilience measures to evaluate how robust a system is to face these problems.

As we will show, the values of the resilience measures stated in this chapter depend on the topology of the communication graph. This graph depends on the applied *partition strategy*, in which the environment is partitioned into sections covered separately by individual robots. Partitioning or *area decomposition* for path planning in robotics is a widely researched subject in coverage and tracking tasks and a common topology is the grid [49]. In grid based methods, the area partitioning is performed by applying a grid overlay on top of the area leading to a discrete configuration space, where if all the cells are visited, then a complete coverage is assumed (see Figure 3.4). Additionally, cycles are considered in boundary or fence patrolling [36]. A strategy is to fragment the boundary into sections which are patrolled separately by individual robots [33] (see Figure 3.5). Other studied configurations are trees. In fact, in area coverage, often it is assumed that the underlying graph is a tree (computed, for example, on the dual graph of a triangulation of the terrain). Spanning trees have been frequently used for multi-robot coverage [62] and boundary patrolling [37]. Also note that, if we consider the underlying graph G of the area decomposition whose vertices are cells and whose edges are the pairs of close cells then, if G is connected, a spanning tree of G can be used as communication graph because any tree is bipartite and synchronizable (Corollary 2.2.21 and Lemma 2.2.25). In this chapter we also focus on



(a) (b) (c)

Figure 3.5: (a) Solid black stroke represent a border to monitor. The gray strip around the border is the region to cover. (b) Decomposition of the region into cells, one cell per robot and for every cell a closed trajectory to cover it is computed. (c) Simplification of the practical model using an abstraction with circular trajectories. The underlying communication graph is represented using gray dotted strokes.

the problem of computing the resilience of synchronized system for specific cases: when the communication graph is a tree, a grid or a cycle.

3.1 Problem statement

Recall that in this chapter we are going to use the simple model introduced in the previous chapter. Taking into account that the swap operations don't affect the performance of the system we are going to ignore these operations and focus on the study of a synchronized system after the removing of some robots and assuming that no other robot enters the system. Also, recall that in such systems the surviving robots apply a *the shifting strategy*, that is, every time a robot arrives at a link position and the corresponding neighboring robot is not there, it shifts to the neighboring trajectory and continues with the schedule of the new trajectory.

Definition 3.1.1 (Synchronized communication system (SCS))**.** *Let* $T = \{C_1, \ldots, C_n\}$ *be a set of pairwise disjoint unit circles (trajectories). A* synchronized communication system *(SCS) is a team of* n *robots using a synchronization schedule* (α, δ) *with communication graph* $G = (V, E)$ *(which is connected and bipartite) such that* $\delta(C_i) = -\delta(C_j)$ *for all* $\{i, j\} \in E$*. An* m*-partial SCS,* $0 < m \leq n$*, is a synchronized communication system in which* $n - m$ *robots have left the team and the* m *remaining robots apply the shifting strategy.*

Remark 3.1.2. *Note that an SCS is a type of partial SCS where no robots have left. Thus, any claims about partial SCSs holds for SCSs as well. Also, if we have a partial SCS* $\mathcal{F}$ *and a robot leaves* $\mathcal{F}$ *then we get a different partial SCS* $\mathcal{F}'$*, so, every statement on a partial SCS assumes that no more robots leave the system.*

Let us define formally the phenomenons that we are going to study in this chapter.

Definition 3.1.3 (Isolation)**.** *In a partial SCS, a robot is* isolated *or is in* isolation *if every time that it arrives at a link position the corresponding neighbor is not there, causing it to shift to the neighboring trajectory.*

Definition 3.1.4 (Loss of connectivity)**.** *We say that a partial SCS has a* loss of connectivity *if there exists a pair of surviving robots that cannot exchange messages (possibly through a sequence of message relays between neighbors) or if there is only one surviving robot in the system.*

Definition 3.1.5 (Non-covered trajectory section)**.** *In a partial SCS, we say that a section of a trajectory is* not covered *or it is a* non-covered trajectory section *if is not visited by a robot regardless the time the system is running.*

In the following we are going to present the measures to determine the robustness of a SCS to these problems.

Definition 3.1.6 (k-isolation-resilience)**.** *The* k-isolation-resilience *of a SCS (*$k \geq 1$*) is the minimum number* ρ_i*, such that, there exist* ρ_i *robots whose removal causes the isolation of at least* k *surviving robots. If it is not possible to obtain* k *isolated robots then the* k*-isolation-resilience is set to infinity.*

Definition 3.1.7 (Broadcasting resilience)**.** *The* broadcasting resilience *of a SCS is the minimum number* ρ_b*, such that, there exist* ρ_b *robots whose removal causes a loss of connectivity in the system.a set of*

Definition 3.1.8 (Coverage resilience)**.** *The* coverage resilience *of a SCS is the minimum number* ρ_c*, such that, there exist* ρ_c *robots whose removal results in at least one non-covered trajectory section.*

Notice that the coverage resilience is related to the *idle-time* of a point p in a terrain, that is, the maximum time that p is unattended by any of the robots. Notice that the maximum number of robots that can fail so that the idle-time of all points is finite is the value of the coverage resilience minus one. A generalization of the definition above is the t-coverage resilience,

Definition 3.1.9 (t-coverage resilience)**.** *Given* $t > 0$*, the* t*-coverage resilience is the minimum number* ρ_{tc}*, such that, there exist* ρ_{tc} *robots whose removal causes the idle-time of some trajectory point to be at least* t *units of time.*

Remark 3.1.10. *Notice that in a SCS (where no robot has left the system) the idle-time of any trajectory point is 1. So, the* t*-coverage resilience is zero for all* $t \leq 1$*.*

Then, we are going to focus in the following problems:

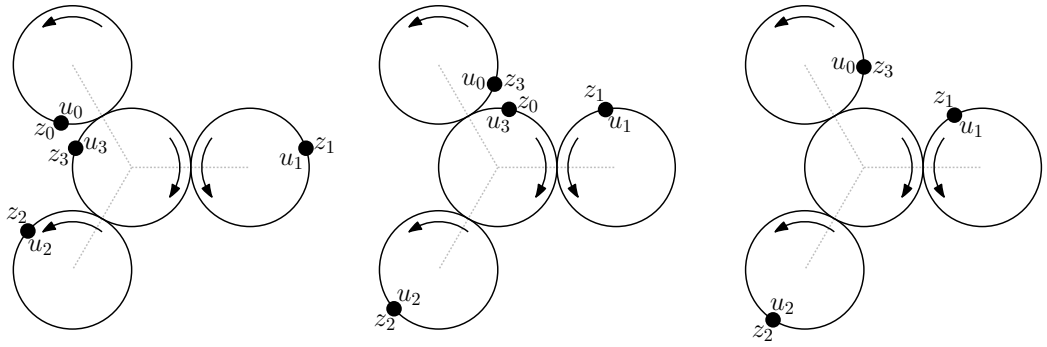


Figure 3.6: Tokens and robots movement. (a) Initial state of the SCS. At time 0, robot u_i holds token z_i. (b) State of the system at time $t_1 > 0$. Robots u_0 and u_3 met and exchanged tokens. (c) State of the system at time $t_2 > t_1 > 0$. Robot u_3 and its token z_0 have been removed.

Problem 3.1.11. *Given a SCS, determine its coverage resilience.*

Problem 3.1.12. *Given a SCS and a natural number k, determine the k-isolation resilience of the system.*

Problem 3.1.13. *Given a SCS, compute its broadcasting resilience.*

Note that higher resilience values correspond to increased fault tolerance. In the next section we are going to introduce some technical tools to tackle the posted problems.

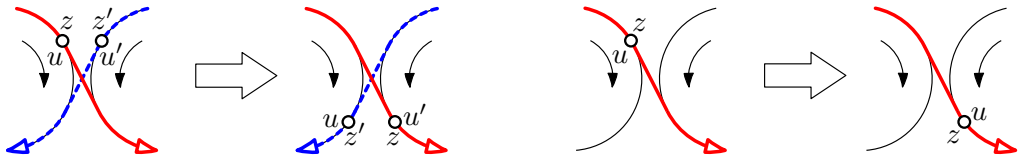
3.2 Technical tools

We are going to introduce the notion of *token* as an abstract entity used to describe the behavior of a partial SCS. Suppose that every robot bears a token when the SCS is deployed. When two robots meet in a communication link, they exchange their tokens. When a robot is removed, the token it carries is removed as well, see Figure 3.6.

A token fulfills the following property:

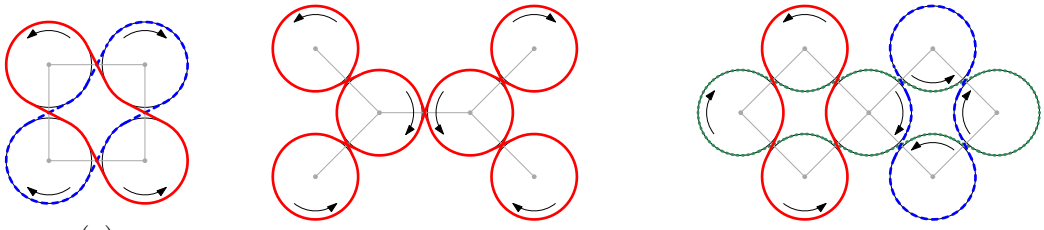
Property 3.2.1. *A token always shifts to the neighboring trajectory at the corresponding link positions.*

Proof. It is pretty obvious that the location of a token in the system is the position of the robot bearing it and they move together at 2π length per time unit speed. Let us focus now on to show that a token always shifts to the neighboring trajectory at the corresponding link positions. Suppose that a robot u bearing a token z arrives at a link position.



(a) (b)

Figure 3.7: (a) Robots u and u', bearing tokens z and z' respectively, arrive at the same time at a link position, so, they meet each other. Analogously, we can say that the tokens z and z' meet each other as well. (b) Robot u, bearing token z, performs a shifting operation because it detects no robot at the link position.



(a) (b) (c)

Figure 3.8: SCSs with two rings (a); one ring (b) and three rings (c).

- If there is a robot u' in the neighboring trajectory then u and u' stay in their current trajectories but they exchange their tokens, so, token z *shifts* to the neighboring trajectory, see Figure 3.7a.
- If there is no robot in the neighboring trajectory then u and z shifts to the neighboring trajectory, see Figure 3.7b.

□

To tackle this problem we need the notion of a *ring*, first introduced in [23]. In the sequel, we give useful properties of rings.

Definition 3.2.2 (Ring)**.** *A* ring *in a SCS is the locus of points visited by a token.*

Figure 3.8 shows the rings of three different SCSs.

Remark 3.2.3. *Each ring is a closed path composed of sections of various trajectories and has a direction of travel determined by the movement direction in the participating trajectories. Each section of a trajectory between two consecutive link positions participates in exactly one ring, thus, the rings in a SCS are pairwise disjoint.*

Remark 3.2.4. *The form of a ring or the number of rings in a system do not depend on the set of active robots, that is, the rings remains invariant in a system regardless of the removal robots. The rings only depend on the topology of the communication graph of the system and the travel directions assigned to the trajectories (notice that inverting the travel directions of the trajectories we get rings with the same form but with opposite travel directions).*

From the definition of ring and Property 3.2.1 we derive:

Property 3.2.5. *The position of a token is the position of the robot who is bearing it. A token always remains in its initial ring and travels with constant speed 2π length per time unit along its ring.*

Remark 3.2.6. *Notice that the property above remains invariant regardless of the removal of robots. That is, if a token z is in a ring r of a partial SCS at time t and after some time Δt some robots have been removed but z remains in the resultant partial SCS, then, at time $t + \Delta t$ token z remains in r.*

Definition 3.2.7 (Path in a ring)**.** *A* path *in a ring r from a point $p \in r$ to a point $q \in r$ is the ordered set of visited points from p to q following the travel direction of r (it may contain tours on r). If a path does not contain any tour in the ring then we say that it is a* simple path.

As suggested by the examples in Figure 3.8, the lengths of rings in a system varies from ring to ring. In discussing the length of a ring, it is convenient to ignore the effect on distance arising from shifting between neighboring trajectories, i.e., to proceed as if neighboring circular trajectories were tangent to each other.

Definition 3.2.8 (Length of a ring)**.** *The* length of a ring *is defined as the sum of the lengths of the trajectory arcs forming the ring. Analogously, the* length of a path in a ring *is defined as the sum of the lengths of the trajectory arcs (as many times as they are traversed) forming the path.*

Figure 3.9 illustrates the above definitions. The following proposition is a technical result needed to describe the length of a simple path between two robots (or tokens) in the same ring.

Proposition 3.2.9. *Let (α, δ) be the synchronization schedule used in a partial SCS with set of trajectories $T = \{C_1, \ldots, C_n\}$. Let σ be a path in a ring r. Let $A_1, \ldots, A_s$ denote the directed arcs traversed in σ when following the travel direction of r, and let C_i denote the trajectory containing A_i.*[2] *Then, for all $1 \leq$*

[2]Note that a path between two robots may have two arcs in the same trajectory, see Figure 3.9c for an example. Therefore, distinct indexes i and j may exist such that $C_i = C_j$. This does not affect the proof of the claim because it is not required that the arcs belong to different trajectories.

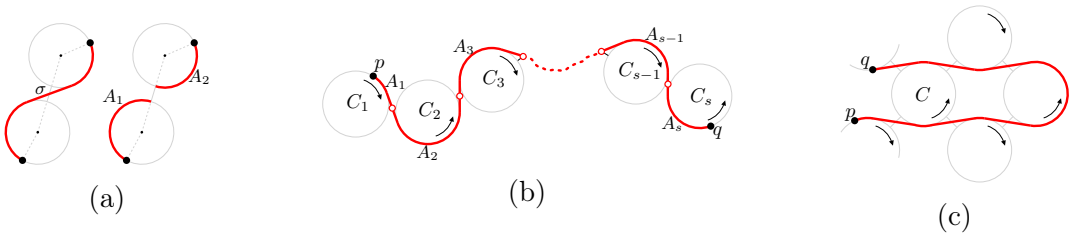


Figure 3.9: (a) The length of a section σ of a ring is the sum of the lengths of the arcs A_1 and A_2. (b) Path from position p to position q in the same ring. The second endpoint of A_i and the first endpoint of A_{i+1} are marked with a common hollow point for all $1 \leq i < s$. (c) A simple path between two robots in a ring that has two arcs in the trajectory C.

$j \leq s$,

$$\alpha(C_j) + \delta(C_j) \cdot 2\pi \cdot (t + \sum_{i=1}^{j} t_i) = \theta_j. \tag{3.1}$$

where t_i is the required time by a robot to traverse A_i and θ_j is the angle position of the second endpoint of A_j in C_j.

Proof. We prove equation (3.1) by induction on j. For $j = 1$, by equation (2.1), we have the base case:

$$\alpha(C_1) + \delta(C_1) \cdot 2\pi \cdot (t + t_1) = \theta_1.$$

Inductive hypothesis. Suppose that equation (3.1) holds for some $j < s$. By the definition of the synchronization schedule we have

$$\alpha(C_{j+1}) + \delta(C_{j+1}) \cdot 2\pi \cdot \left(t + \sum_{i=1}^{j} t_i \right) = \theta'_{j+1},$$

where θ'_{j+1} is the position of the first endpoint of A_{j+1} in C_{j+1}. Since $\theta_{j+1} = \theta'_{j+1} + \delta(C_{j+1}) \cdot 2\pi \cdot t_{j+1}$, we have

$$\alpha(C_{j+1}) + \delta(C_{j+1}) \cdot 2\pi \cdot \left(t + \sum_{i=1}^{j+1} t_i \right) = \theta_{j+1},$$

and the claim follows. □

The following lemma is established in [23] with a slightly different argument.

Lemma 3.2.10. *In a partial SCS, let p and q be the positions on trajectories C_i and C_j at time t, respectively. If p and q are in the same ring r, then the length of the path between p and q is in $2\pi\mathbb{N}$.*

Proof. Let (α, δ) be the synchronization schedule of the system. Then:

$$\begin{aligned} p &= \alpha(C_i) + \delta(C_i) \cdot 2\pi \cdot t \text{ and,} \\ q &= \alpha(C_j) + \delta(C_j) \cdot 2\pi \cdot t. \end{aligned} \tag{3.2}$$

Let σ be the path in the ring from p to q following the travel direction on r. Let t_σ be the required time to travel σ. Then:

$$\begin{aligned} \alpha(C_j) + \delta(C_j) \cdot 2\pi \cdot (t + t_\sigma) &= q \text{ (by Proposition 3.2.9)} \\ \alpha(C_j) + \delta(C_j) \cdot 2\pi \cdot t + \delta(C_j) \cdot 2\pi t_\sigma &= q \\ q + \delta(C_j) \cdot 2\pi t_\sigma &= q \text{ (by equation 3.2))} \\ \delta(C_j) \cdot 2\pi t_\sigma &= 0 \end{aligned}$$

Therefore, the angle $2\pi t_\sigma$ is in $2\pi\mathbb{Z}$. Since $2\pi t_\sigma$ is the length of the path σ, the lemma follows. □

Corollary 3.2.11. *In a partial SCS, the length of a path between two robots (tokens) in the same ring is in* $2\pi\mathbb{N}$.

Proof. Let t be an arbitrary instant of time. Let p and q the positions at time t of two robots (tokens) in the same ring. Then, using Lemma 3.2.10 the result follows. □

Corollary 3.2.12. *The length of every ring in a SCS is in* $2\pi\mathbb{N}$.

Proof. Let r be an arbitrary ring. Let p be a point of r. Let σ be the path that starts at p and follows the travel direction of r until p is reached again. Notice that r and σ have the same length and, by Lemma 3.2.10 the length of σ is in $2\pi\mathbb{N}$. □

Lemma 3.2.13. *In a partial SCS the number of robots in a given ring remains invariant. If the length of the ring is* $2l\pi$ *then it has at most* l *robots. Furthermore, in a SCS (where no robots have left the system), a ring of length* $2\ell\pi$ *has exactly* ℓ *robots, each at distance* 2π *from the next.*

Proof. From Property 3.2.5 we directly derive that: in a partial SCS the number of tokens in a ring remains invariant. So, having into account that a token is carried by a robot the first claim follows. From Corollary 3.2.11 and Corollary 3.2.12 we deduce that a ring of length $2l\pi$ has at most l robots. We now prove the third claim. Consider a system of n trajectories and m rings. Suppose that the i-th ring has length $2l_i\pi$ and x_i robots, $1 \leq i \leq m$. Then $x_i \leq l_i$, for all i, and $n = \sum_{i=1}^{m} x_i$. Since the rings are disjoint, $\sum_{i=1}^{m} l_i = n$. Then

$$n = \sum_{i=1}^{m} x_i \leq \sum_{i=1}^{m} l_i = n,$$

and we conclude that $l_i = x_i$ for all i. □

Two tokens *meet* each other if they arrive at the same communication link at the same time (see Figure 3.7a, i.e., the robots bearing the tokens arrive at the same communication link at the same time).

The proof of the following result follows easily from Property 3.2.5:

Lemma 3.2.14. *Let z and z' be two tokens of a partial SCS in rings r and r' (possibly the same ring), respectively. The tokens z and z' will meet each other if and only if there is a communication link ℓ where the rings r and r' cross each other, and, there are two paths of the same length L: one from z to ℓ in r (possibly longer than r) and other from z' to ℓ in r' (possibly longer than r').*

Definition 3.2.15 (Correspondence between tokens)**.** *Let z and z' be two tokens in a partial SCS. We say that there is a* correspondence *between z and z' at time t if there exists a $\triangle t > 0$ such that z and z' meet each other at time $t + \triangle t$.*

Let x be a point in a ring r and let d be a non-negative real number. The point $x + d$ is the point reached by traveling distance d from x following the travel direction in r. Analogously, the point $x - d$ is the point y such that $y + d$ is x. Notice that d could be greater than the length of r. Also notice that $x - d + d = x + d - d = x$.

Lemma 3.2.16. *Let r and r' be two rings (could be the same ring) of a partial SCS of lengths $2\pi\mu$ and $2\pi\mu'$, respectively. Let z and z' be two tokens in r and r', respectively. If z and z' meet each other at a communication link ℓ then they will meet each other at ℓ every $\mathrm{mcm}(\mu, \mu')$ (minimum common multiple of μ and μ') time units.*

Proof. Suppose that z and z' meet each other at ℓ at time t. The location reached after traveling $\mathrm{mcm}(\mu, \mu')$ time units from ℓ in r is the point $\ell + 2\pi \cdot \mathrm{mcm}(\mu, \mu') = \ell$. The same occurs by traveling on r'. Therefore, at time $t + \mathrm{mcm}(\mu, \mu')$ the tokens z and z' will meet each other at ℓ. The result follows from successive applications of this argument. □

Lemma 3.2.17. *Let z and z' be two tokens in a partial SCS at time t_1. Suppose that at time $t_2 > t_1$ a set (possibly empty) of robots (with their respective tokens) have been removed from the system but z and z' remain in the resultant partial SCS. Then, there is a correspondence between z and z' at time t_1 if and only if there is a correspondence between z and z' at time t_2.*

Proof. Let r and r' be the rings of z and z', respectively (r and r' could be the same ring). Let $2\pi\mu$ and $2\pi\mu'$ be the lengths of r and r', respectively.

Suppose there is a correspondence between z and z' at time t_1. Let x_{t_1} and x'_{t_1} be the positions of z and z' in r and r', respectively, at time t_1. Then, by the definition of correspondence and Lemma 3.2.14, there are paths p and p' (in r and

r' respectively) of length $2\pi\triangle t$ starting at x_{t_1} and x'_{t_1}, respectively, and ending at a communication link ℓ (where r and r' cross each other). That is:

$$x_{t_1} + 2\pi\triangle t = \ell$$
$$x'_{t_1} + 2\pi\triangle t = \ell.$$

Let k be the smallest non-negative integer such that $k \cdot \text{mcm}(\mu, \mu') + \triangle t \geq t_2 - t_1$. From Lemma 3.2.16, we have that:

$$x_{t_1} + 2\pi(k \cdot \text{mcm}(\mu, \mu') + \triangle t) = \ell$$
$$x'_{t_1} + 2\pi(k \cdot \text{mcm}(\mu, \mu') + \triangle t) = \ell.$$

Let $x_{t_1} + 2\pi(t_2 - t_1)$ and $x'_{t_1} + 2\pi(t_2 - t_1)$ be the positions occupied by z and z' after traveling for $t_2 - t_1$ time units, respectively. Let q and q' be the paths obtained by traveling from these positions in r and r', respectively, during $k \cdot \text{mcm}(\mu, \mu') + \triangle t - (t_2 - t_1)$ time units. Notice that q and q' have the same length $2\pi(k \cdot \text{mcm}(\mu, \mu') + \triangle t - (t_2 - t_1))$. Moreover:

$$x_{t_1} + 2\pi(t_2 - t_1) + 2\pi(k \cdot \text{mcm}(\mu, \mu') + \triangle t - (t_2 - t_1)) = \ell$$
$$x'_{t_1} + 2\pi(t_2 - t_1) + 2\pi(k \cdot \text{mcm}(\mu, \mu') + \triangle t - (t_2 - t_1)) = \ell.$$

Therefore, there will be a correspondence between z and z' at time t_2.

Now, suppose there is a correspondence between z and z' at time t_2. Let x_{t_2} and x'_{t_2} be the positions at time t_2 of z and z' in r and r', respectively. Then, by definition of correspondence and Lemma 3.2.14, there are two paths p and p' (in r and r' respectively) of length $2\pi\triangle t$ starting at x_{t_2} and x'_{t_2}, respectively, and ending at a link position ℓ. That is:

$$x_{t_2} + 2\pi\triangle t = \ell$$
$$x'_{t_2} + 2\pi\triangle t = \ell.$$

Then, from Property 3.2.5 and Remark 3.2.6 we have that $x_{t_2} - 2\pi(t_2 - t_1)$ and $x'_{t_2} - 2\pi(t_2 - t_1)$ were the positions at time t_1 of z and z', respectively. Let q and q' be the paths obtained by traveling from these positions in r and r', respectively, during $t_2 - t_1 + \triangle t$ time units. Notice that q and q' have the same length $2\pi(t_2 - t_1 + \triangle t)$. Moreover:

$$x_{t_2} - 2\pi(t_2 - t_1) + 2\pi(t_2 - t_1 + \triangle t) = x_{t_2} + 2\pi\triangle t = \ell$$
$$x'_{t_2} - 2\pi(t_2 - t_1) + 2\pi(t_2 - t_1 + \triangle t) = x'_{t_2} + 2\pi\triangle t = \ell.$$

Therefore, there was a correspondence between z and z' at time t_1 and the result follows. □

Remark 3.2.18. *The previous lemma states that the* relation *of correspondence between two tokens z and z' remains invariant regardless of the removal of robots unless z or z' is removed from the system.*

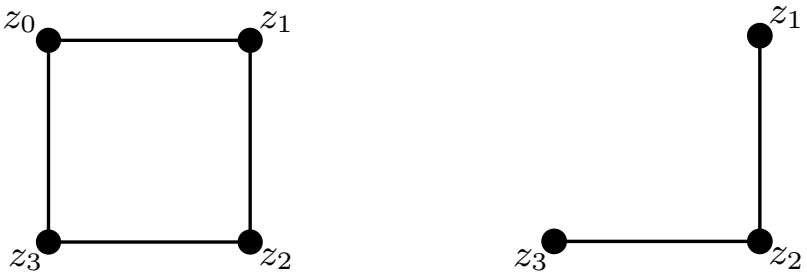


Figure 3.10: (a) Meeting graph of the SCS of Figure 3.6 at time 0 (Figure 3.6a). Notice that the meeting graph remains invariant at time t_1 (Figure 3.6b). (b) The meeting graph of the resultant partial SCS after removing robot u_3 (Figure 3.6c).

Definition 3.2.19 (Meeting graph)**.** *Let $\mathcal{F}$ be an m-partial SCS. Let Z be the set of surviving tokens. The* meeting graph *of $\mathcal{F}$ is the graph $\mathcal{T}_\mathcal{F}$ whose vertices are the tokens in Z (that is, $V(\mathcal{T}_\mathcal{F}) = Z$) and whose set of edges is:*

$$E(\mathcal{T}_\mathcal{F}) = \{\{z, z'\} | z \in Z, z' \in Z, \text{there is a correspondence between } z \text{ and } z'\}.$$

From Remark 3.2.18, the meeting graph of a partial SCS remains invariant while no additional robots are removed. Figure 3.10a shows the meeting graph of the SCS of Figure 3.6 before the robot u_3 is removed. From the definition of meeting graph and Lemma 3.2.17, the next result (illustrated in Figure 3.10b) follows:

Lemma 3.2.20. *Let $\mathcal{F}$ be an m-partial SCS with set of tokens Z. Let $\mathcal{F}'$ be the m'-partial SCS resulting from the removal of some robots from $\mathcal{F}$. Let $Z' \subset Z$ be the set of tokens of $\mathcal{F}'$. Let $\mathcal{T}_\mathcal{F}$ and $\mathcal{T}_{\mathcal{F}'}$ denote the meeting graphs of $\mathcal{F}$ and $\mathcal{F}'$ respectively. Then, $\mathcal{T}_{\mathcal{F}'}$ is the subgraph of $\mathcal{T}_\mathcal{F}$ induced by Z'.*

3.3 Computing coverage resilience

In this section we show how to compute the coverage resilience for arbitrary graphs as well as for trees and grids.

Notice that if a token z is at a point x of a ring r then x is being covered by the robot bearing z. This simple observation allows us to study the coverage resilience using tokens.

Upon deployment of a system, a ring of length $2\pi l$ contains l tokens at distance 2π from one to the next (Lemma 3.2.13). From Property 3.2.5 and Remark 3.2.6, the distribution of the tokens in a ring when some robots have been removed from the system looks like Figure 3.11. The following theorems are deduced:

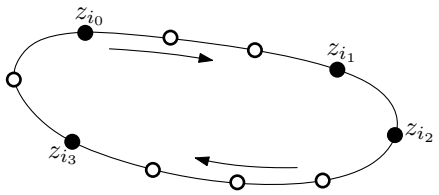


Figure 3.11: Tokens in a ring of a partial SCS. The hollow points represent the removed tokens.

Lemma 3.3.1. *Let r be a ring of length $2l\pi$ in a partial SCS. If $a < l$ is the maximum number of absent consecutive tokens in r then the idle-time of any point in r is $a + 1$ (see Figure 3.11).*

Lemma 3.3.2. *Let r be a ring in a partial SCS. A trajectory segment of r is non-covered if and only if r does not contain surviving robots.*

From Remark 3.2.3 and Theorem 3.3.2 the following result is deduced:

Corollary 3.3.3. *Let c be the number of rings in an SCS. Let $2l_1\pi, 2l_2\pi, \ldots, 2l_c\pi$ be the lengths of the c rings. The coverage resilience of the system is the minimum of $\{l_1, l_2, \ldots, l_c\}$.*

Theorem 3.3.4. *The coverage resilience of a given SCS can be computed in linear time on the number of trajectories in the system.*

Proof. Corollary 3.3.3 suggests a simple algorithm to compute the coverage resilience of a SCS by determining the rings in the system and their lengths. To do that, it is sufficient to follow the movement of a token until reaching the starting point. In this way, the length of a detected ring is the sum of the lengths of the traversed arcs. The complexity of detecting a ring in this way is $O(m)$ where m is the number of link positions traversed by the ring. When a ring is detected (by returning to the starting point), then we choose a non visited trajectory arc in order to detect another ring and so on. If there are no more non-visited trajectory arcs then we are done. Let G be the communication graph of the SCS. Recall that every link position of the system is an edge of the communication graph G. Clearly, the complexity of this algorithm is $O(|E|)$ where E is the set of edges in G. Taking into account that the communication graph is planar then this algorithm has running time $O(n)$ where n is the number of trajectories. □

Given a value $t > 0$, the t-coverage resilience can also be computed using the rings of the SCS. Let c be the number of rings in the system and $2l_1\pi, 2l_2\pi, \ldots, 2l_c\pi$ their lengths. Let l^* be the minimum of $\{l_1, l_2, \ldots, l_c\}$. Using Lemma 3.3.1 we deduce that if $l^* \geq \lceil t \rceil$, then the t-coverage resilience is $\lceil t \rceil - 1$; otherwise, the t-coverage resilience is l^*. As a consequence, we can state the following:

Theorem 3.3.5. *The t-coverage resilience of an SCS is $\min\{l^*, \lceil t \rceil - 1\}$ where $2\pi l^*$ is the length of the shortest ring in the system.*

(a) (b)

Figure 3.12: (a) Ring corresponding to T'. (b) Ring corresponding to T.

3.3.1 Coverage resilience for trees and grids

Trees

In this section we are going to focus in computing the coverage resilience for some specific cases of the communication graph: trees and grids. Our goal is to improve the linear time algorithm proposed in the proof of Theorem 3.3.4 by using that the communication graph of the system is a tree or a grid.

Lemma 3.3.6. *If the communication graph of a SCS is a tree then there is a single ring.*

Proof. We prove the lemma by induction on the number of trajectories. Clearly, if there is only one trajectory, the ring is unique.

Suppose that the claim holds for any tree with n trajectories. We show that it also holds for any tree T with $n+1$ trajectories. Let C be a trajectory corresponding to a leaf in T, see Figure 3.12a. Let T' be the tree obtained by deleting trajectory C. Then there is exactly one ring corresponding to T'. Adding C to the system, the ring changes by adding a loop covering C as shown in Figure 3.12b and the lemma follows. □

From the previous lemma, Corollary 3.3.3 and Theorem 3.3.5, it is directly derived that:

Corollary 3.3.7. *If the communication graph of a SCS with n trajectories is a tree then, the coverage resilience and the t-coverage resilience can be computed in constant time. Moreover, the coverage resilience of the system is n and the t-coverage resilience is* $\min\{n, \lceil t \rceil - 1\}$.

Therefore the system is very stable (with respect to covering) because regardless the number of removed robots, the remaining robots (if there are any) will cover all the trajectories.

Grids

In the rest of this section, we study the coverage resilience of a SCS where the communication graph is a grid. Consider a set of $M \cdot N$ trajectories distributed

in M rows and N columns. Each trajectory is identified by a pair (i, j) where $1 \leq i \leq M$ and $1 \leq j \leq N$ indicate the row and the column, respectively, where the trajectory is located. In the communication graph, trajectory (i, j) is linked to trajectories $(i-1, j)$ if $i > 1$, $(i, j-1)$ if $j > 1$, $(i+1, j)$ if $i < M$ and $(i, j+1)$ if $j < N$. We refer to this type of SCS as a *grid SCS*. An *m-partial grid SCS* is analogously defined.

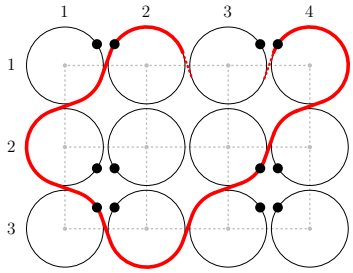


Figure 3.13: A 3×4 grid SCS. The drawn portion of a ring hits the top boundary at trajectories $(1, 2)$ and $(1, 4)$, it hits the bottom boundary at $(3, 2)$, the left boundary at $(2, 1)$ and the right one at $(1, 4)$.

In a grid SCS we say that a ring *hits* the top boundary of the grid if the ring passes through the top section of a circle in the first row. Analogously we can define when a ring hits the left, bottom or right boundary of the ring. See Figure 3.13.

Figure 3.14 shows the local behavior of a ring as it visits the trajectories of a grid SCS. In this Figure, small white squares denote two kinds of points: the points where the ring hits the boundaries and the communication links between two neighboring circles (for simplicity, tangent trajectories are considered). Note that these points are traversed diagonally (with slopes 1 and -1) by a ring. We consider the *movement lattice* formed by these points (see Figure 3.15). If the grid communication graph has M rows and N columns then the movement lattice has $2M+1$ rows indexed from 0 (topmost) to $2M$ (bottommost) and $2N+1$ columns indexed from 0 (leftmost) to $2N$ (rightmost). In this way, every vertex of the lattice can be referenced by its row and column (see Figure 3.15). Notice that a token moves diagonally on this lattice (following the ring that houses it) and only

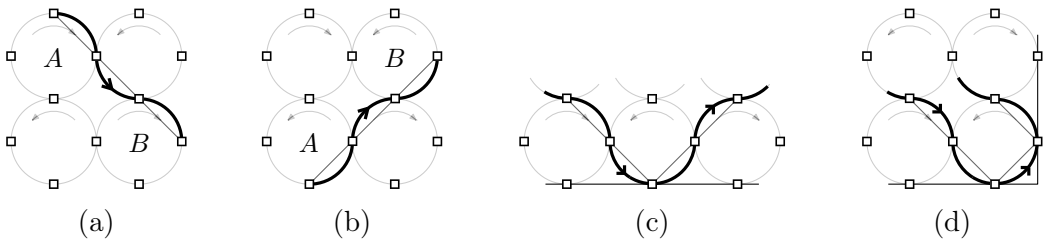


Figure 3.14: Local behavior of a ring in a grid SCS.

changes its direction when reaching a vertex with out-degree equal to one (i.e. when it hits a grid boundary, it *bounces*).

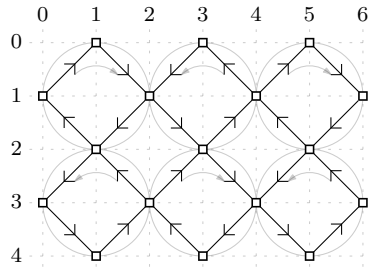


Figure 3.15: Movement lattice in a grid.

Lemma 3.3.8. *An $M \times M$ grid SCS has M rings of length $2M\pi$. Each ring hits each of the four boundaries exactly once.*

Proof. Let H be the movement lattice of the system. Let r be the ring that hits the top boundary at the vertex $(0, i)$ of H. It is easy to see that r hits the right boundary at vertex $(2M-i, 2M)$, the bottom boundary at the vertex $(2M, 2M-i)$ and the left boundary at $(i, 0)$. Also, r does not hit the boundaries in any other vertex. Note that a step from one vertex to another in the movement lattice corresponds to a section of a ring of length $\pi/2$. The ring r visits $4M$ vertices on H, thus the length of r is $4M \cdot \pi/2$, i.e., $2M\pi$. Repeating this argument for each hitting point in the top boundary we obtain M rings of length $2M\pi$. The sum of the lengths of all circles in the system is $2M^2\pi$ and the sum of the lengths of the M rings in the system is $2M^2\pi$ too, so there is no other ring in the system. □

Let S_1 and S_2 be two grid SCSs. We say that S_1 and S_2 are *concatenable* if S_1 and S_2 have the same number of rows M and for all $1 \leq i \leq M$ the movement direction assigned to the last trajectory of the i-th row of S_1 is opposite to the movement direction assigned to the first trajectory of the i-th row of S_2. The *concatenation* of S_1 and S_2, such that the last trajectory in the i-th row of S_1 is linked with the first trajectory in the i-th row of S_2 (for all $1 \leq i \leq M$), produces a new $M \times (N + N')$ grid SCS, see Figure 3.16.

The following result is a technical lemma that we need in order to complete the proof of Theorem 3.3.10.

Lemma 3.3.9. *Let S and U be two concatenable grid SCSs of size $M \times N$ and $M \times M$, respectively. Suppose that S has k rings of the same length l, and every ring in S hits each of the left and right boundaries exactly c times, and hits each of the top and bottom boundaries exactly c' times. Let R be the $M \times (N + M)$ grid SCS resulting from the concatenation of S and U. Then, R has exactly k rings of the same length $l + 2cM\pi$ and every ring in R hits each of the left and right boundaries exactly c times, and hits each of the top and bottom boundaries exactly $c' + c$ times.*

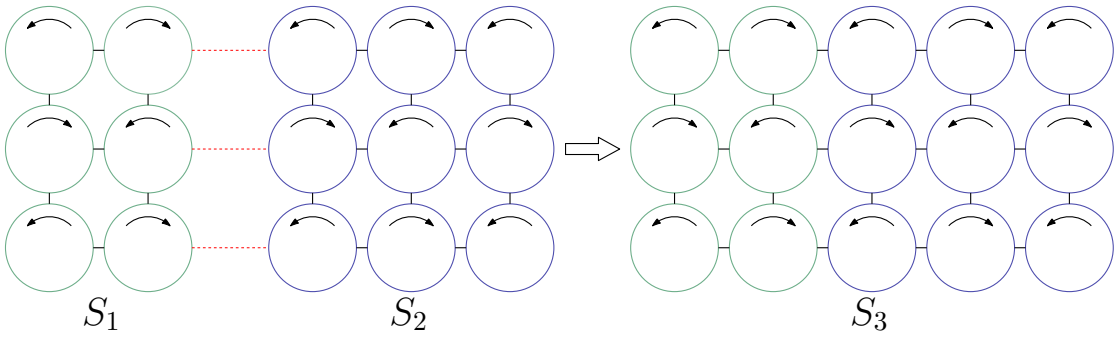


Figure 3.16: S_3 is the resultant SCS of the concatenation of the SCSs S_1 and S_2.

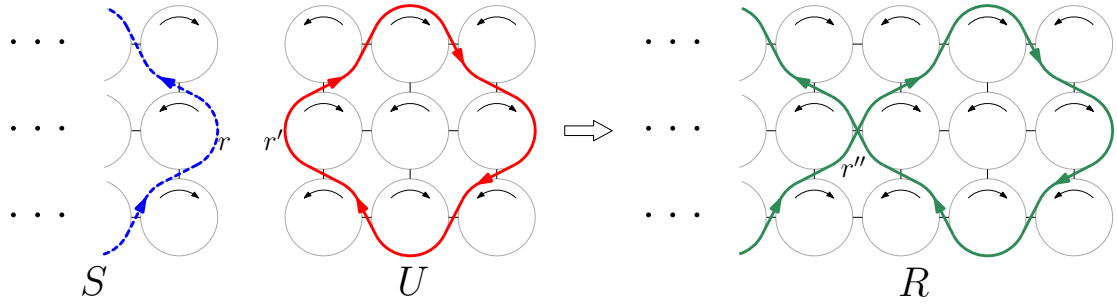


Figure 3.17: r is the ring in S that hits the right boundary in the second row. r' is the ring in U that hits the left boundary in the second row. r'' is the ring in R obtained from r and r' in the concatenation of S and U.

Proof. By Lemma 3.3.8, U has M rings of length $2M\pi$, and all of them hit each boundary once. Thus, every ring in U extends the length of a ring in S by $2M\pi$, see Figure 3.17. Since every ring in S hits the right boundary c times then, after concatenation, each ring in S is fused with c rings in U. Note that two rings in S can not be fused together in the concatenation. Then, every ring in R is formed by the fusion of one ring in S and c rings in U. Therefore, R has k rings of the same length $l + 2cM\pi$. Every ring so obtained hits the left boundary c times at the c hitting points of the respective ring in S, and hits the right boundary c times at the hitting points of the c respective rings in U. Every ring in R hits each of the top and bottom boundaries $c' + c$ times, at the c' hitting points of the respective ring in S and at the hitting points of the c respective rings in U. □

We now proceed to establish a key result of this subsection.

Theorem 3.3.10. *An $M \times N$ grid SCS has $\gcd(M, N)$ (greatest common divisor of M and N) rings of the same length $\frac{2\pi MN}{\gcd(M,N)}$. Moreover, every ring hits each of the left and right boundaries $\frac{M}{\gcd(M,N)}$ times and hits each of the top and bottom boundaries $\frac{N}{\gcd(M,N)}$ times.*

Proof. We prove the result by induction in the number of rows. For $M = 1$, we have that every $1 \times N$ grid SCS has a single ring, $\gcd(1, N) = 1$, one hitting point in left and right boundaries, and N hitting points in the top and bottom boundaries. Thus, the theorem holds in this case. Assume as inductive hypothesis that: for a fixed value P, the theorem holds for every $M \times N$ grid SCS with $M \leq P$.

We need to prove the theorem for a $(P + 1) \times N$ grid SCS. If $N \leq P$ then, using the fact that a $(P+1) \times N$ grid SCS is equivalent to a $N \times (P+1)$ grid SCS, the theorem holds by the inductive hypothesis. If $N = P + 1$ then the theorem holds by Lemma 3.3.8. In order to prove the theorem for a $(P + 1) \times N$ grid SCS with $N > P + 1$ we use induction in the number of columns. Assume as second inductive hypothesis that: for a fixed value $Q \geq P+1$, the theorem holds for every $(P + 1) \times N$ grid SCS with $N \leq Q$.

Let S be a $(P + 1) \times (Q + 1)$ grid SCS. We have that $Q + 1 > P + 1$, then removing the last $P + 1$ columns of S we obtain a $(P + 1) \times (Q - P)$ grid SCS denoted by S'. The theorem holds for S' by the second inductive hypothesis. The $P+1$ removed columns conform a $(P+1) \times (P+1)$ grid SCS which is concatenable with S'. So, concatenating S' with the $P+1$ removed columns we obtain S. Then, by using Lemma 3.3.9 and properties of the greatest common divisor, the result follows. □

By using Theorem 3.3.10, Corollary 3.3.3 and Theorem 3.3.5 we arrive to the main result of this subsection:

Theorem 3.3.11. *The coverage resilience of an $M \times N$ grid SCS is:*

$$\frac{M \cdot N}{\gcd(M, N)}.$$

And, the value of the t-coverage resilience is:

$$\min\left\{\frac{M \cdot N}{\gcd(M, N)}, \lceil t \rceil - 1\right\}.$$

Corollary 3.3.12. *The coverage resilience and t-coverage resilience of an $N \times M$ grid SCS can be computed in $O(T_{gcd}(N, M))$[3] time.*

3.4 Computing k-isolation resilience

In this section we are going to prove that the problem of computing the k-isolation resilience of a SCS is NP-hard when k is part of the input. Also, we are going to show that the corresponding decision problem is NP-complete. Additionally we are going to study this problem for small values of k.

[3] $T_{gcd}(N, M)$ denotes the required time to compute $\gcd(N, M)$. Taking into account that $n = N * M$ then, $T_{gcd}(N, M) = O(\log n(\log\log n)^2 \log\log\log n)$ according to [129].

To conclude this section we study the problem of computing the k-isolation resilience when communication graph is a tree, a cycle or a grid.

The following lemma relates isolation with tokens and will be useful for computing the k-isolation resilience.

Lemma 3.4.1. *Let u be a robot in a partial SCS $\mathcal{F}$. Let z be the token carried by u. The following statements are equivalent:*

- *u is isolated.*
- *u always bears z.*
- *z has degree zero in the meeting graph of $\mathcal{F}$.*

Proof. If u is isolated then it never meet another robot in the communication links, so, it does not exchange its token and bears z forever.

Let us prove now that if u always bears z, then z has degree zero in the meeting graph of $\mathcal{F}$. For the sake of contradiction, assume that there is a token z' such that there is a correspondence between z and z'. When z meets z' if u is bearing z then u exchanges tokens with the robot bearing z', so, after the meeting, u is bearing z'. If when z meets z' the token z is not carried by u then, u has exchanged tokens previously. In any case there is a contradiction.

If z has degree zero in the meeting graph of $\mathcal{F}$ then z never meets another token, so, u never meets another robot, i.e., u is isolated.

□

From the previous lemma we can say that a robot is isolated if it is bearing an *isolated* token.

3.4.1 Hardness of computing the k-isolation resilience

The following notion gives us a useful tool to address the hardness of computing the k-isolation resilience of a SCS.

Definition 3.4.2 (Isolation number)**.** *The* isolation number *of a SCS is the maximum possible number of isolated robots (tokens) in a partial SCS.*

The following result is deduced directly from the definitions of isolation number and k-isolation resilience.

Corollary 3.4.3. *If the isolation number of a SCS is s, then the k-isolation resilience of the system is* infinity *for all $k > s$.*

Lemma 3.4.4. *If the isolation number of a SCS is s then the s-isolation resilience of the system is $n - s$.*

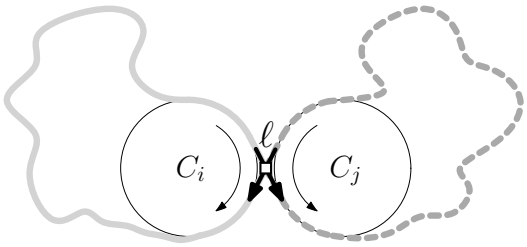


Figure 3.18: Ring that crosses itself at the communication point ℓ. The two travel directions through ℓ are shown with black arrows. The section of the ring on the left (resp. on the right) of ℓ is a tie which is represented using bold light-gray (resp. bold dashed dark-gray) stroke.

Proof. Let R be the s-isolation resilience of the system, by definition $R \leq n - s$. Thus, there exists a set of R robots whose removal induces the isolation of s robots. Suppose $R < n - s$ then, in the resultant partial SCS, the set of non-isolated surviving robots is not empty and its cardinality is greater than 1 by definition of isolation. Therefore the removal of all but one non-isolated surviving robots results in a new partial SCS with $s+1$ isolated robots, a contradiction. □

From Lemma 3.4.1 and the definition of isolation number we have that:

Corollary 3.4.5. *Let $\mathcal{F}$ be a SCS. Let $\mathcal{T}_\mathcal{F}$ be the meeting graph of $\mathcal{F}$. The isolation number of $\mathcal{F}$ is equal to the cardinality of the maximum* independent set[4] *of $\mathcal{T}_\mathcal{F}$.*

We are going to prove that computing the isolation number of a SCS is NP-hard, then, the problem of computing the k-isolation resilience is NP-hard as well.

Let r be a ring in a partial SCS that crosses itself at a communication link ℓ, see Figure 3.18. Notice that ℓ can be traversed in two different directions, one from C_i to C_j and another from C_j to C_i.

Definition 3.4.6 (Tie of a ring)**.** *Let r be a ring in a partial SCS that crosses itself at a communication link ℓ. A* tie *of r is any of the two simple paths that starts and ends at ℓ following the travel direction in r.*

Corollary 3.4.7. *The length of a tie is in $2\pi\mathbb{N}$.*

Proof. Let ℓ be a communication link where a ring r crosses itself between two trajectories C_i and C_j. Let d_{ij} and d_{ji} be travel directions through ℓ determined by r from C_i to C_j and, from C_j to C_i, respectively. Let (α, δ) be the synchronization schedule of the system. Since C_i and C_j are synchronized, there is a value $t > 0$ such that:

$$\alpha(C_i) + 2\pi\delta(C_i)t = \phi_{ij} \text{ and } \alpha(C_j) + 2\pi\delta(C_j)t = \phi_{ji}.$$

[4]Subset of nodes in a graph that does not contain two adjacent nodes.

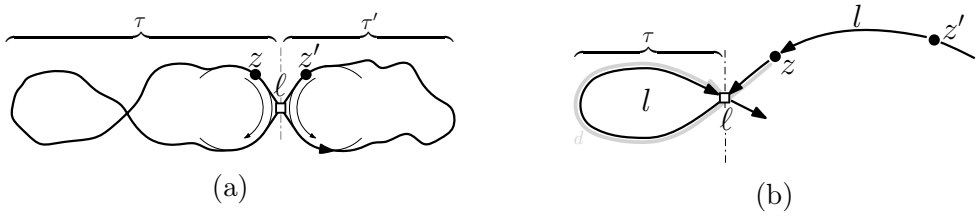


Figure 3.19: (a) The length of the tie τ (resp. τ') is equal to the length of path from z' (resp. z) to z (resp. z') following the travel direction in r. (b) There is a correspondence between tokens z and z' because the distance between them is equal to the length of a tie in the same ring. The bold gray curve from z to ℓ while leaving τ has length d.

Notice that the one of the two ties determined by ℓ can be represented as a simple path that starts at ϕ_{ij} and ends at ϕ_{ji} following the direction d_{ij} (and the other tie would be a simple path that starts at ϕ_{ij} and ends at ϕ_{ji} following the direction d_{ji}). Then, using that the length of a section of r between ϕ_{ij} and ϕ_{ji} is in $2\pi\mathbb{N}$ (Lemma 3.2.10), the result follows. □

From Lemma 3.3.6 and Corollary 3.4.7 we have:

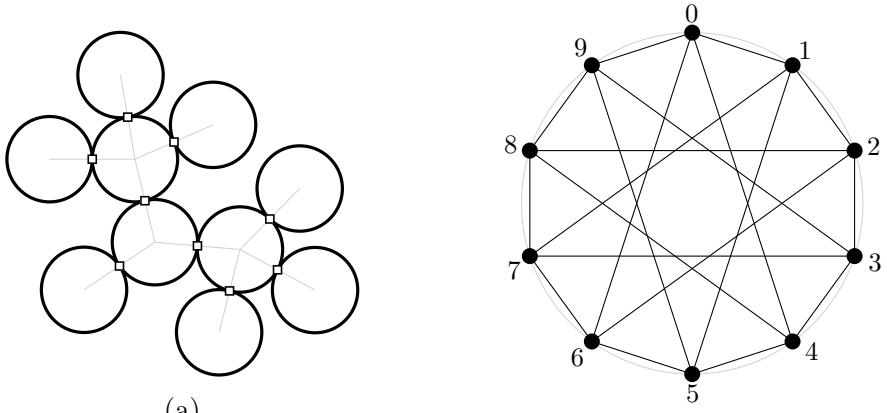
Corollary 3.4.8. *In a SCS of n trajectories whose communication graph is a tree, a communication link determines two ties of lengths $2l\pi$ and $2(n-l)\pi$ respectively, where $l \in \mathbb{N}$.*

Lemma 3.4.9. *Let z and z' be two tokens on a ring r of an m-partial SCS. There is a correspondence between z and z' if and only if r has a tie whose length is equal to the length of a simple path between z and z'.*

Proof. ($\Rightarrow$) Suppose there is a correspondence between z and z'. Then z and z' must meet each other after a while at a communication link ℓ where r crosses itself. Observe that the path from z' to z following the travel direction in r is one of the ties determined by ℓ, see Figure 3.19a, so the length of this tie is equal to the length of the path from z' and z.

($\Leftarrow$) Suppose r has a tie, let us say τ, whose length is equal to the length l of the path from z' to z following the travel direction in r. Let ℓ be the communication link that determines τ, see Figure 3.19b. There are two ways to reach ℓ, one entering and the other leaving τ. Let d be the length of the path from z to ℓ leaving τ. The point obtained by traveling d units of length from the current positions of z and z', respectively, is ℓ in both cases, z reaches ℓ leaving τ and z' reaches ℓ entering τ. Consequently, there is a correspondence between z and z'. □

From lemmas 3.4.9, 3.3.6, and 3.4.1 we deduce:



(a) (b)

Figure 3.20: (a) Example of a SCS whose communication graph is a tree. Notice that the length of every tie is in $2\pi \cdot \{1, 4, 6, 9\}$. (b) The circulant graph $C_{10}\{1, 4\}$.

Corollary 3.4.10. *In an m-partial SCS whose communication graph is a tree, a token z is isolated if and only if the distance between z and any other live token is different from the lengths of all ties in the single ring of the system.*

We now show that SCSs whose communication graphs are trees are related to *circulant graphs*. A graph on n nodes is *circulant* if the nodes of the graph can be numbered from 0 to $n-1$ such that, if two nodes x and $(x+d) \bmod n$ are adjacent, then two nodes z and $(z+d) \bmod n$ are adjacent for any z. We call such a node numbering a *c-order*.

Remark 3.4.11. *Let $G = (V, E)$ be a circulant graph of n vertices. The* set of jumps *of G is $J = \{\min\{(i-j) \bmod n, (j-i) \bmod n\} \mid (i,j) \in E\}$. Notice that for all $(i,j) \in E$ we have that $i + d = j$ or $j + d \equiv i \pmod n$ for some $d \in J$. Then, the circulant graph G can be encoded by the number of vertices n and the set of jumps J, and, it can be shortly denoted as $C_n J$. Notice that $1 \leq d \leq \lfloor \frac{n}{2} \rfloor$ for all $d \in J$. As an illustration, see Figure 3.20b where the circulant graph $C_{10}\{1, 4\}$ is shown.*

Lemma 3.4.12. *Let $\mathcal{F}$ be a SCS of n trajectories whose communication graph is a tree. The meeting graph $\mathcal{T}_\mathcal{F}$ of $\mathcal{F}$ is a circulant graph.*

Proof. Let r be the single ring in $\mathcal{F}$. Let $0, 1, \ldots, n-1$ be a circular enumeration of the tokens in r, following the travel direction of the ring. Lemma 3.2.13 implies that token i is 2π units ahead of token $i-1 \pmod n$, as usual). Let $2l_1\pi, \ldots, 2l_t\pi$ be the lengths of all ties in r. Let $\mathcal{T}_\mathcal{F}$ be the meeting graph of $\mathcal{F}$. By Lemma 3.4.9, the tokens adjacent to token i in $\mathcal{T}_\mathcal{F}$ are $\{i + l_1, \ldots, i + l_t\}$. Then, $\mathcal{T}_\mathcal{F}$ is circulant. □

Figure 3.20a shows a SCS with a ring of length 20π. Notice that the length of every tie is 2π, 8π, 12π or 18π. Therefore, enumerating the tokens of this ring from

0 to 9 in the travel direction of the ring, we see that token i has correspondences with tokens $i+1, i+4, i+6$ and $i+9$, (mod 10). Figure 3.20b shows the meeting graph of this system, which is the circulant graph $C_{10}\{1,4\}$.

In the following, we define an auxiliary operation to transform a circulant graph into another circulant graph with some interesting properties for us.

Definition 3.4.13 ($K_{n,n}$-augmentation)**.** *Let $G = (V, E)$ be a graph with n nodes. A graph $G' = (V', E')$ is a* clone *of G if G' and G are isomorphic and $V \cap V' = \emptyset$. The* $K_{n,n}$-augmentation *of G, denoted by $\overline{G} = (\overline{V}, \overline{E})$, is the graph resulting from a graph join operation between G and a clone G', i.e $\overline{V} = V \cup V'$ and $\overline{E} = E \cup E' \cup \{\{v, w\} \mid v \in V, w \in V'\}$.*

From now on we denote a vertex in a graph of n vertices by v_i with $i \in \{0, \ldots, n-1\}$. In general, the vertex indices are taken modulo n.

The following result can be deduced directly from the definition of circulant graphs.

Lemma 3.4.14. *A graph is circulant if and only if all its connected components are isomorphic to the same circulant graph.*

Proof. ($\Leftarrow$) Let $G = (V, E)$ be a circulant graph of n nodes and let $v_0, \ldots, v_{n-1}$ be a c-order of G. Construct a graph G' as the union of m pairwise disjoint clones of G and let $v_0^{(i)}, \ldots, v_{n-1}^{(i)}$ be the c-order of i-th clone corresponding to the c-order of G. It is easy to see that

$$v_0^{(1)}, v_0^{(2)}, \ldots, v_0^{(m)}, v_1^{(1)}, v_1^{(2)}, \ldots, v_1^{(m)}, \ldots, v_{n-1}^{(1)}, v_{n-1}^{(2)}, \ldots, v_{n-1}^{(m)}$$

is a c-order of G'. Therefore G' is circulant.

($\Rightarrow$) Let $G = (V, E)$ be a circulant graph and let $v_0, \ldots, v_{n-1}$ be its c-order. Let C be a connected component of G containing v_0. Let $m > 0$ be the lowest value such that $v_m \in C$. Then the nodes $v_{2m}, v_{3m}, \ldots$ are in C. It can be proven that m divides i for all $v_i \in C$. It can also be proven that m divides n. Therefore $C = \{v_0, v_m, v_{2m}, \ldots, v_{n-m}\}$. From here, it is easy to see that G has m isomorphic connected components of the form $\{v_i, v_{i+m}, v_{i+2m}, \ldots, v_{n-m+i}\}$ for all $0 \leq i < m$. □

Lemma 3.4.15. *Let $G = (V, E)$ and $\overline{G} = (\overline{V}, \overline{E})$ be a graph and its $K_{n,n}$-augmentation, respectively. G is a circulant graph if and only if $\overline{G}$ is a circulant graph.*

Proof. Let $G' = (V, E)$ be the clone of G used in the creation of $\overline{G}$.

($\Rightarrow$) Let $v_0, \ldots, v_{n-1}$ be a c-order of G and $v'_0, \ldots, v'_{n-1}$ be the corresponding c-order of G'. Consider the ordering $L = (v_0, v'_0, v_1, v'_1, \ldots, v_{n-1}, v'_{n-1})$ of $\overline{V}$. We show that L is a c-order of $\overline{G}$. Indeed, if d is odd then, for any i, (v_i, v_{i+d}) is an

(a) (b)

Figure 3.21: (a) Circulant graph $C_6\{2\}$. (b) Circulant graph $C_{12}\{1,3,4,5\}$ which is the $K_{6,6}$-augmentation of $C_6\{2\}$, solid points denote the nodes of the original graph, and non-solid ones, the vertices of its clone. Original and cloned vertices are connected with dashed edges.

edge of $\overline{G}$. If d is even, then both v_i and v_{i+d} are in the same graph G or G'. Then $(v_i, v_{i+d}) \in \overline{E}$ if and only if $(v_0, v_{i+d/2}) \in E$. Therefore the graph $\overline{G}$ is circulant.

($\Leftarrow$) If $\overline{G}$ is circulant then its complement graph $\neg\overline{G}$ is circulant. By Lemma 3.4.14, $\neg\overline{G}$ has m isomorphic components. Since $\neg\overline{G}$ is the union of $\neg G$ and $\neg G'$, $\neg G$ has $m/2$ isomorphic components that are circulant graphs. Thus, G is a circulant graph by Lemma 3.4.14. □

Lemma 3.4.16. *Let $G = (V, E)$ and $\overline{G} = (\overline{V}, \overline{E})$ be a graph and its $K_{n,n}$-augmentation, respectively. The maximum independent set of G and the maximum independent set of $\overline{G}$ have the same cardinality.*

Proof. Let $H \subseteq V$ and $\overline{H} \subseteq \overline{V}$ be maximum independent sets in G and $\overline{G}$, respectively. Notice that the vertices in H also form an independent set in $\overline{G}$, thus $|H| \leq |\overline{H}|$. Since $\overline{G}$ is the $K_{n,n}$-augmentation, $\overline{H}$ cannot contain a vertex from V and a vertex from V'. So, either $\overline{H} \subseteq V$ or $\overline{H} \subseteq V'$. Then $|\overline{H}| \leq |H|$ and $|H| = |\overline{H}|$. □

Let C_nJ be a circulant graph. Let $C_{2n}\overline{J}$ denote the $K_{n,n}$-augmentation of C_nJ where

$$\overline{J} = \{2d \mid d \in J\} \cup \left\{ 2i - 1 \;\middle|\; 1 \leq i \leq \left\lfloor \frac{n+1}{2} \right\rfloor \right\}.$$

Figure 3.21 shows an example of a circulant graph and its $K_{n,n}$-augmentation. Notice that the set of jumps of the $K_{n,n}$-augmentation of C_nJ contains all the odd numbers in the interval $[1, n]$.

We are ready to prove the main result of this section.

Theorem 3.4.17. *The problem of computing the isolation number of a SCS (IN-SCS) is NP-hard, even, if the communication graph of the SCS is a* caterpillar tree[5].

[5]A caterpillar tree is a tree in which all the vertices are within distance 1 of a central path.

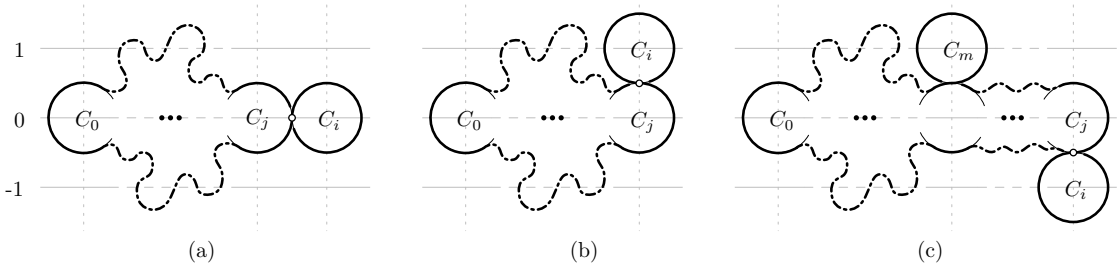


Figure 3.22: Addition of C_i to the SCS. The ring of the SCS is shown with bold stroke. The case of $i \in \overline{J}$ is shown in (a). The case of $i \notin \overline{J}$ is shown in (b) and (c). If i is the smallest value not in $\overline{J}$ then add C_i to the 1-line (b). If it isn't, then let $m < i$ be the greatest value such that C_m is not centered on the 0-line. If C_m is centered on the -1-line then C_i is added to the 1-line (b). If C_m is centered on the 1-line then C_i is added to the -1-line (c).

Proof. We use a reduction from the problem of computing the maximum independent set in a circulant graph (MIS-CG) which is NP-hard [32]. Let C_nJ be a circulant graph with $n \geq 2$, as input to the MIS-CG problem. For convenience we work with $C_{2n}\overline{J}$ which is the $K_{n,n}$-augmentation of the given circulant graph C_nJ. Recall that the problem of computing a maximum independent set for C_nJ is equivalent to the problem of computing a maximum independent set for $C_{2n}\overline{J}$ (Lemma 3.4.16) and that $\overline{J}$ contains all odd numbers in $[1, n]$.

By Corollary 3.4.5, it suffices to transform $C_{2n}\overline{J}$ into a SCS of $2n$ circles whose communication graph is a caterpillar tree such that

$$d \in \overline{J} \quad \text{if and only if} \quad \text{there is a tie of length } 2d\pi \text{ in the SCS.} \tag{3.3}$$

We place the circles on three horizontal lines with coordinates in 1, 0 and -1 as illustrated in Figure 3.22. First, place the circle C_0 on the 0-line. Then place $C_i, i = 1, \ldots, n$ as follows. Let C_j be the last circle placed on the 0-line.

1. If $i \in \overline{J}$ then add the circle C_i to the 0-line touching C_j, see Figure 3.22a.

2. If $i \notin \overline{J}$ then add the circle C_i touching C_j but alternating between centered on the 1-line and centered on the -1-line. In other words, if the last added circle not centered on the 0-line is centered on the 1-line, then center C_i on the -1-line, and vice-versa. see Figures 3.22b and 3.22c, respectively.

Notice that i in the second case is even since $\overline{J}$ contains all odd numbers in $[1, n]$. Thus, the next circle C_{i+1} will be placed on the 0-line. Since the lines 1 and -1 alternate, C_i touches only one circle, C_j, which is placed in the 0-line.

We have placed $n + 1$ circles $C_0, \ldots, C_n$. In order to add the $n - 1$ remaining circles we proceed as follows:

- if n is even then:

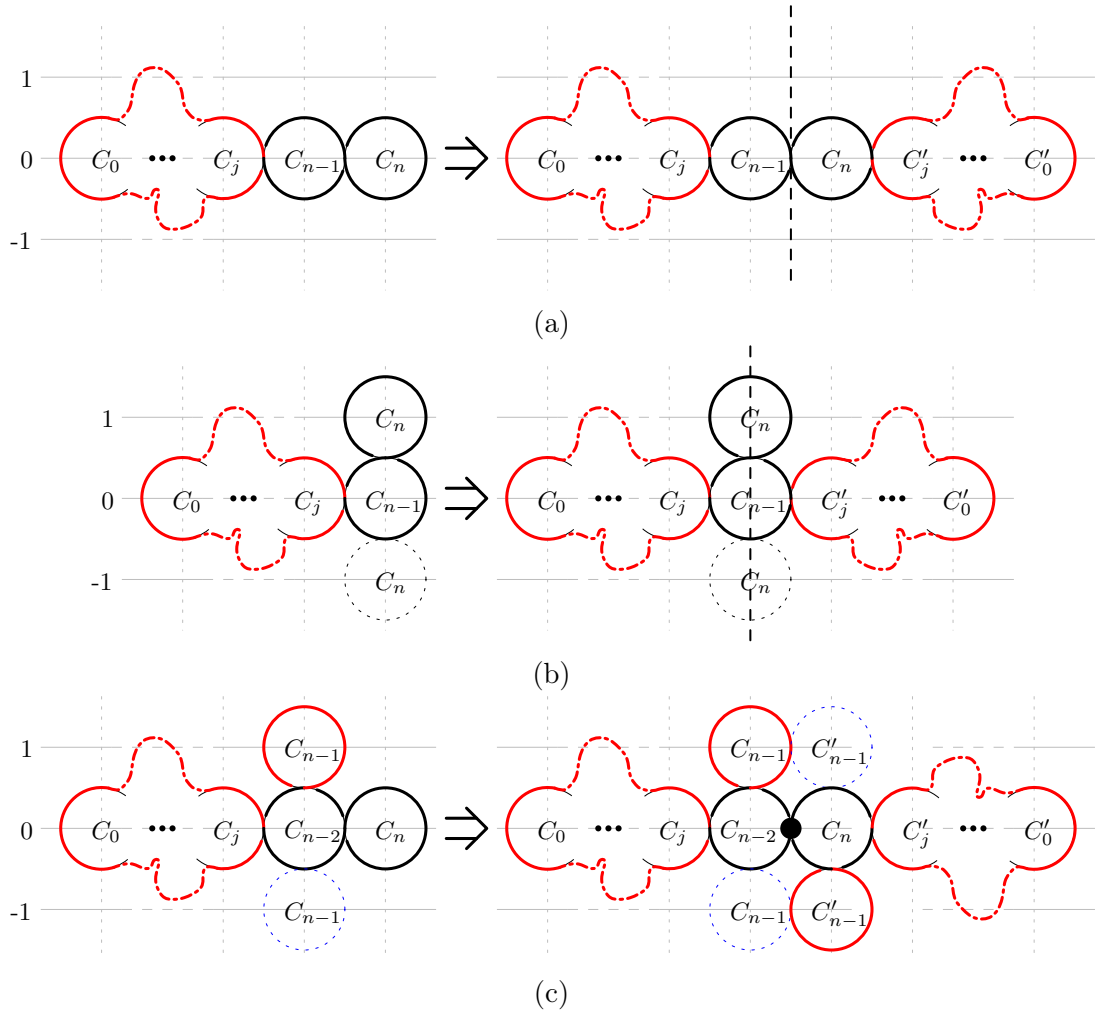


Figure 3.23: Instructions of how to add the $n-1$ remaining circles. (a) If C_{n-1} and C_n are both on the 0-line, then apply symmetry about the vertical line between C_{n-1} and C_n. (b) If C_{n-1} is on the 0-line but C_n is not, then apply symmetry about the vertical line passing through the center of C_{n-1}. (c) In the remaining case apply symmetry about the touching point of C_{n-2} and C_n.

  - if $n \in \overline{J}$ then we proceed as shown in Figure 3.23a.
  - if $n \notin \overline{J}$ then we proceed as shown in Figure 3.23b.

- if n is odd then:
  - if $(n-1) \in \overline{J}$ then we proceed as shown in Figure 3.23a.
  - if $(n-1) \notin \overline{J}$ then we proceed as shown in Figure 3.23c.

Now, we are ready to prove statement (3.3) on the obtained SCS.

($\Rightarrow$) If $d \in \overline{J}$, then C_d is centered on the 0-line and the tie determined by the crossing point between C_d and the previous circle centered on the 0-line covers the d circles to the left of C_d. Consequently, the length of this tie is $2d\pi$.

($\Leftarrow$) Every crossing point between circles centered on the same vertical line determines two ties of length 2π and $2(2n-1)\pi$, respectively, and 1 is in $\overline{J}$. Consider the crossing point between two circles centered on the 0-line. By symmetry, we can assume that the circles are C_j and C_i where $0 \leq j < i \leq n$. The crossing point determines a tie of length $2i\pi$ covering the i circles to the left of C_i. Since C_i is on the 0-line, $i \in \overline{J}$. This argument completes the proof.

Figure 3.24 shows some examples of the SCS construction. □

The following result is deduced from Theorem 3.4.17 and Lemma 3.4.4.

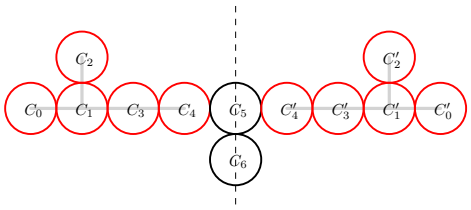
Corollary 3.4.18. *The problem of computing the k-isolation resilience of a SCS is NP-hard.*

In the rest of this section we focus on how to count the number of isolated robots (tokens) in an m-partial SCS in order to prove that the decision version of k-isolation resilience problem is NP-Complete.

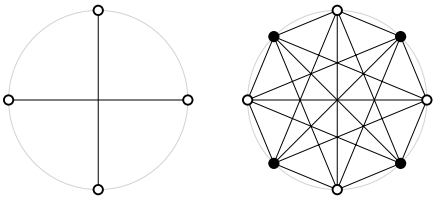
Lemma 3.4.9 gives us a method to check if there is a correspondence between two tokens of the same ring. But, how does one check if there is a correspondence between two tokens of different rings?

Theorem 3.4.19. *In an m-partial SCS, let z and z' be two tokens in different rings r and r', respectively. Let ℓ be communication link where r and r' cross each other. Let d and d' denote the lengths of the simple paths from z and z' to ℓ at an arbitrary time instant t, respectively. Let $2l\pi$ and $2l'\pi$ be the lengths of r and r' respectively. Then:*

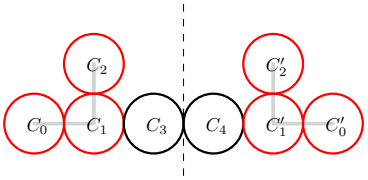
- *$d - d'$ is in $2\pi\mathbb{Z}$.*
- *Let $s \in \mathbb{Z}$ such that $d - d' = 2\pi s$. Then, there is a correspondence between tokens z and z' if and only if the* greatest common divisor *of l and l' divides s.*



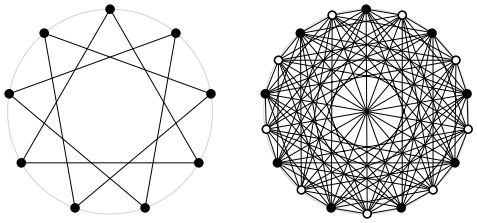
(a) SCS obtained from the circulant graph $C_{12}\{1, 3, 4, 5\}$ shown in Figure 3.21b.



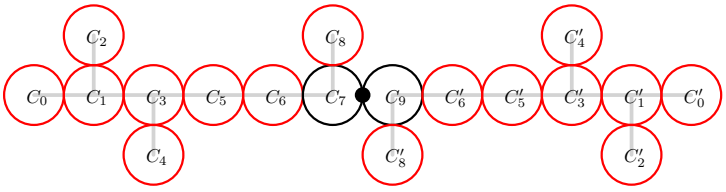
(b) $C_4\{2\}$ and its $K_{4,4}$-augmentation: $C_8\{1, 3, 4\}$.



(c) SCS obtained from $C_8\{1, 3, 4\}$.



(d) $C_9\{3\}$ and its $K_{9,9}$-augmentation: $C_{18}\{1, 3, 5, 6, 7, 9\}$.



(e) SCS obtained from $C_{18}\{1, 3, 5, 6, 7, 9\}$.

Figure 3.24: Examples of construction of a SCS from the $K_{n,n}$-augmentation of some circulant graphs. The samples in (a) and (c) are obtained by applying the steps illustrated in Figure 3.23b and 3.23a, respectively. The example in (e) is obtained by applying the steps illustrated in Figure 3.23c.

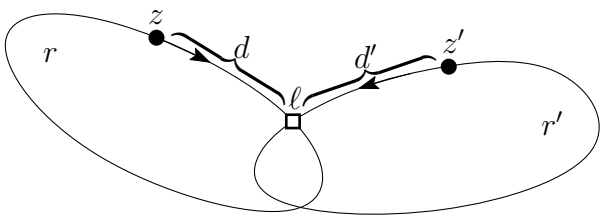


Figure 3.25: Illustration of Theorem 3.4.19.

Proof. Figure 3.25 shows the positions of the tokens. First, we prove that $(d-d') \in 2\pi\mathbb{Z}$. Suppose, w.l.o.g., that $d \leq d'$. Consider the time $t + \Delta t$ when z reaches ℓ. At time t the distance from z' to ℓ is $d' - d$. Due to the synchronization, at time $t + \Delta t$ there is or there should be (in case it has been removed) a token of r' at ℓ. Then, using Lemma 3.2.10 we get that $d' - d \in 2\pi\mathbb{N}$.

We focus now on the second claim. If there is a correspondence between z and z' then at anarbitrary time instant t there are two paths of equal length, let us say L, from z and z' to ℓ. The path from z (resp. z') to ℓ can be decomposed in the section from z (resp. z') to ℓ and zero or more round trips in r (resp. r'). Let x and y denote the number of round trips of the paths in r and r', respectively. Therefore $L = d + x \cdot 2\pi l$ and $L = d' + y \cdot 2\pi l'$ where $(x, y) \in \mathbb{N} \times \mathbb{N}$. Then:

$$\begin{aligned} d + x \cdot 2\pi l &= d' + y \cdot 2\pi l' \\ d - d' &= y \cdot 2\pi l' - x \cdot 2\pi l \\ 2\pi s &= y \cdot 2\pi l' - x \cdot 2\pi l \\ s &= y \cdot l' - x \cdot l \end{aligned} \tag{3.4}$$

Considering x and y variables, this is a *Diophantine equation* and has a solution where x and y are integers if and only if $\gcd(l, l') \mid s$.

($\Rightarrow$) If there is a correspondence between z and z', then there exists a solution $(x_1, y_1) \in \mathbb{N} \times \mathbb{N}$ for equation (3.4). Therefore, $\gcd(l, l') \mid s$.

($\Leftarrow$) If $\gcd(l, l') \mid s$ then there exist infinitely many solutions for equation (3.4) in $\mathbb{Z} \times \mathbb{Z}$. Observe that the line represented by equation (3.4) has positive slope. Therefore, there exist infinite solutions in $\mathbb{N} \times \mathbb{N}$. Taking any of these solutions in $\mathbb{N} \times \mathbb{N}$, we can obtain two paths, one from z to ℓ and the other from z' to ℓ of equal lengths. Therefore, there is a correspondence between z and z'. □

Given an m-partial SCS, the problem of counting how many of the m robots (tokens) are isolated takes polynomial time. This can be done using a *correspondence test* between every pair of live tokens z and z' in the system.

> **Correspondence test:** *Given two tokens z and z', is there a correspondence between them?*

Lemma 3.4.20. *Given an m-partial SCS, the* correspondence test *between two tokens z and z' in the system takes $O(nT_{gcd}(n))$ time, where $T_{gcd}(n)$ is the time*[6] *to compute the greatest common divisor between any two numbers less than or equal to n.*

Proof. The first thing to do is a preprocessing in order to find the rings of the system and their lengths. During this process we can also obtain the ties and their lengths. This preprocessing step takes $O(|E|)$ time where E is the set of edges of the communication graph. Since this graph is planar,the preprocessing step takes $O(n)$ time. After that, we proceed as follows in order to perform the *correspondence test* between every pair of tokens in the system. Let z and z' be two tokens in the system. If z and z' are in the same ring, the test can be done in $O(n)$ time by checking ties of the ring (Lemma 3.4.9). If they are in different rings, r and r' respectively, then there are two options. (i) If r and r' have no common crossing points then there is no a correspondence between them. (ii) Otherwise, for every communication link ℓ where r and r' cross each other check if z and z' meet each other at ℓ using Theorem 3.4.19. This can be done in $O(T_{gcd}(n))$ time, where $T_{gcd}(n)$ is the time to compute $gcd(l, l')$ and $2l\pi$ and $2l'\pi$ are the lengths of rings r and r', respectively. Thus, the *correspondence test* takes $O(nT_{gcd}(n))$ time. □

Corollary 3.4.21. *Given an m-partial SCS, the total time to count the number of isolated robots (tokens) is $O(m^2nT_{gcd}(n))$.*

As a consequence of the above results, we arrive at the following result:

Corollary 3.4.22. *The decision problems: determining if the k-isolation resilience of a SCS is smaller than a given value s and determining if the isolation number of a SCS is greater than a given value s are both NP-complete, even if the communication graph is a caterpillar tree.*

3.4.2 Algorithm to compute *k*-isolation resilience

Despite the problem of computing the k-isolation resilience of SCS is NP-hard we may want to compute its values. In this subsection we present an algorithm to compute this resilience measure in $O(kn^{k+1})$ time, so, this algorithm can be useful for small values of k.

Lemma 3.4.23. *Constructing the meeting graph of a SCS takes $O(n^2)$ time.*

Proof. Let $\mathcal{F}$ be a SCS. Let Z be the set of tokens. In $O(n)$ time we can discover all the rings, the ties and the position of the tokens in the rings. To build the

[6] $T_{gcd}(n) = O(\log n(\log\log n)^2 \log\log\log n)$ according to [129].

meeting graph $\mathcal{T}_\mathcal{F}$ we start by setting $V(\mathcal{T}_\mathcal{F}) = Z$. Then, to conform $E(\mathcal{T}_\mathcal{F})$ do the following:

Same ring: For each token z in a ring r, find tokens in r at positions $p + l_1, p + l_2, \ldots$ where p is the position of z and $\{l_1, l_2 \ldots\}$ is the set of the lengths of the ties in r. Add edges to $E(\mathcal{T}_\mathcal{F})$ between z and every found token. The running time of this step is $O(n)$ for one token z and $O(n^2)$ in total.

Distinct rings: For each communication link ℓ where two different rings r and r' cross each other. Compute $g = gcd(l, l')$, where $2l\pi$ and $2l'\pi$ are the lengths of r and r' respectively. For each token z in r at distance d from ℓ, find all tokens in r' at distance $d + i \cdot g, i = 0, 1, 2, \ldots$. Add edges between z and these tokens to $E(\mathcal{T}_\mathcal{F})$. This can be done in $O(n + I) = O(n)$ time per communication link where I is the number of edges found for communication link. The total time is $O(n^2)$. □

Let $\mathcal{T}_\mathcal{F}$ be the meeting graph of a SCS $\mathcal{F}$. In the following we describe an algorithm to compute the k-isolation resilience of $\mathcal{F}$ by exploring all the possible k independent sets of $\mathcal{T}_\mathcal{F}$.

Theorem 3.4.24. *Computing the k-isolation resilience of a SCS takes $O(kn^{k+1})$ time.*

Proof. The idea to compute the k-isolation resilience is the following:
In a preprocessing step compute the meeting graph $\mathcal{T}_\mathcal{F}$ of the SCS in $O(n^2)$ time. Now, let A be a dynamic set with the available (surviving) tokens. At the beginning $A = \{z_1, \ldots, z_n\}$ contains all n tokens. In general, if A is empty, we cannot select a new token and we check another set of k tokens. When a token z_i is selected from A, we remove its neighbors in $\mathcal{T}_\mathcal{F}$ from A. If z_i is not the last (k-th selected) token, then A contains candidate tokens to select next. If z_i is the last selected token, then $|A|$ is the number of remaining tokens. Let A_{max} be the largest set A of remaining tokens over all selections of k tokens (if it is not possible to select k tokens then the k-isolation resilience is ∞). Then $n - k - |A_{max}|$ is the k-isolation resilience of the system.

The number of analyzed subsets is $O(n^k)$ (all the possible subsets of k tokens) and every subset is analyzed in $O(nk)$ time. Then, the result follows. □

3.4.3 Improvement on computing 1-isolation resilience

Remark 3.4.25. *Let $\mathcal{F}$ be a SCS. Let $Z = \{z_1, \ldots, z_n\}$ be the initial set of tokens in $\mathcal{F}$. Let $N(z)$ denote the number of tokens that have a correspondence with z, $z \in Z$. By definition, the value of 1-isolation resilience is $\min\{N(z) | z \in Z\} = \delta(\mathcal{T}_\mathcal{F})$ where $\delta(\mathcal{T}_\mathcal{F})$ denotes the degree of the vertex with minimum degree in the meeting graph $\mathcal{T}_\mathcal{F}$ of $\mathcal{F}$.*

Computing the 1-isolation resilience of a SCS using the algorithm of Theorem 3.4.24 takes $O(n^2)$ time. The following theorem improves this complexity for this specific case.

Theorem 3.4.26. *The 1-isolation resilience of a SCS can be computed in $\tilde{O}(n) = O(nT_{gcd}(n))$ time.*

Proof. In a preprocessing step, find the rings and their lengths, also, obtain the ties and their lengths. This preprocessing takes $O(n)$ time. For each ring r, pick a single token z from it. Let $r_1, r_2, \ldots$ be the rings crossing r. For each ring r_i crossing r, compute $g_i = gcd(l, l_i)$ in $O(T_{gcd}(n))$ time where $2l\pi$ and $2l_i\pi$ are the lengths of r and r_i, respectively. Also, compute a list D_i in the following way: D_i starts empty, then, for each communication link ℓ where r_i crosses r, let d be the distance from z to ℓ. If there is a value $d' \in D_i$ such that g_i divides $d - d'$ then D_i stays the same, else, add d to D_i. Compute $N(z) = t(r) + \sum_i |D_i| \cdot l_i/g_i$ where $t(r)$ is the distinct lengths of ties in r. This is the number of tokens that have a correspondence with z (Lemma 3.4.9 and Theorem 3.4.19). Let x be another token laid $2b\pi$ behind z in r. Notice that if a token z' in a ring r' has a correspondence with z, then the token x' laid $2b\pi$ behind z' in r' has a correspondence with x. From this observation we have that for every two tokens z and x in the same ring, $N(x) = N(z)$. Let $\varrho(r)$ be the value $N(z)$ of an arbitrary token z in the ring r. Computing $\varrho(r)$ takes $O(e_r \cdot T_{gcd}(n))$ time where e_r is the number of communication links traversed by r. Find the smallest $\varrho(r)$ for all rings r in the system and this is the 1-isolation resilience value.

The total running time of this algorithm is $O(nT_{gcd}(n)) = \tilde{O}(n)$[7] because every communication link is analyzed two times, one per each traversing ring (the same ring twice in ties) and the number of communication links is $O(n)$. □

3.4.4 *k*-isolation resilience for trees, cycles and grids

This subsection focuses on computing the k-isolation resilience for some specific cases of the communication graph: cycles and grids. We propose polynomial time algorithms for the cases of cycles or grids. We also improve the results of sections 3.4.2 and 3.4.3 when the communication graph is a tree.

Trees

Lemma 3.4.27. *Computing the set J of jumps of the circulant meeting graph of a SCS whose communication graph is a tree takes linear time.*

Proof. From Lemma 3.4.9 and Corollary 3.4.8 we have that traveling in the system along the ring, we can determine in linear time the length of all the ties in the

[7] $\tilde{O}$ notation hides polylogarithmic factors.

system. Moreover, from properties of the circulant graphs, Lemma 3.4.12 and Corollary 3.4.8, we have that $l \in J$ if and only if there is a tie of length $2l\pi$ and $l \leq n/2$. The result follows. □

From the previous lemma, Remark 3.4.25 and properties of the circulant graphs, the following result is straightforward deduced:

Lemma 3.4.28. *In a SCS whose communication graph is a tree, the value of the 1-isolation resilience is* $2|J|$*, if* $n/2 \notin J$ *and,* $2|J|-1$*, in other case, and it can be computed in linear time.*

Now, let us focus on computing the 2-isolation resilience of a SCS whose communication graph is a tree.

Lemma 3.4.29. *Let* $L = \{l_1, \ldots, l_t\}$ *be the set of tie lengths in the single ring of a SCS whose communication graph is a tree. Let* S_+ *and* S_- *be the multisets:*

$$S_+ = \{l_j + l_i \mid (l_i, l_j) \in L \times L,\ l_j + l_i \notin L,\ l_j + l_i < n\} \text{ and,}$$
$$S_- = \{l_j - l_i \mid (l_i, l_j) \in L \times L,\ l_j - l_i \notin L,\ l_j - l_i > 0\}.$$

Then the 2-isolation resilience of the system is $2t - f$*, where* f *is the frequency of the* mode[8] *in* $S_+ \cup S_-$*.*

Proof. Let $z_0, z_1, \ldots, z_{n-1}$ be a circular labeling of the tokens such that there is a correspondence between two tokens z_i and z_j if and only if $((j-i) \mod n) \in L$. W.l.o.g. assume that z_0 and z_d are two isolated tokens after removing a set R of tokens with minimum cardinality (i.e., $|R|$ is the 2-isolation resilience). Notice that $0 < d < n$ and $d \notin L$. Let $S_0 = \{z_l \mid l \in L\}$ denote the set of tokens having a correspondence with z_0. By other hand, let $S_d = \{z_{d+l} \mid l \in L\}$ (the subindices $d + l$ are taken modulo n) denote the set of tokens having a correspondence with z_d. Notice that $R = S_0 \cup S_d$ and $|R| = 2t - |S_0 \cap S_d|$ which is the 2-isolation resilience of the system. The tokens having a correspondence with both z_0 and z_d are in $S_0 \cap S_d$. Let us prove that $|S_0 \cap S_d|$ is equal to the frequency f_d of d in $S_+ \cup S_-$.

Firstly we prove that $f_d \leq |S_0 \cap S_d|$, that is, every occurrence of d in $S_+ \cup S_-$ induces a different token in $S_0 \cap S_d$.

- For every occurrence of d in S_-, there is a pair $(l_i, l_j) \in L \times L$ such that $l_j - l_i = d$. Then, $l_j = d + l_i$ and $z_{l_j} \in S_d$. Clearly, $z_{l_j} \in S_0$, and therefore, $z_{l_j} \in S_0 \cap S_d$. Also, for every two different occurrences of d in S_- there are two different pairs $(l_i, l_j) \in L \times L$ and $(l_{i'}, l_{j'}) \in L \times L$ such that $l_j - l_i = l_{j'} - l_{i'} = d$. It is easy to check that $l_j \neq l_{j'}$. Then, $z_{l_j} \neq z_{l_{j'}}$ and both are in $S_0 \cap S_d$.

[8]The mode of a multiset is the value that appears most often.

- For every occurrence of d in S_+ there is a pair $(l_i, l_j) \in L \times L$ such that $l_i + l_j = d$. Notice that $(n - l_i) \in L$ (by properties of circulant graphs) and $((d + (n - l_i)) \mod n) = l_j$. Then, $z_{l_j} \in S_d$. Clearly $z_{l_j} \in S_0$, thus, $z_{l_j} \in S_0 \cap S_d$. Analogously to the previous case, for every two different occurrences of d in S_+, there are two different pairs $(l_i, l_j) \in L \times L$ and $(l_{i'}, l_{j'}) \in L \times L$ such that $l_j + l_i = l_{j'} + l_{i'} = d$. It is easy to check that $l_j \neq l_{j'}$. Then, $z_{l_j} \neq z_{l_{j'}}$ and both are in $S_0 \cap S_d$. Finally, notice that, if there is a pair $(l_i, l_j) \in L \times L$ such that $l_j + l_i = d$ (resp. $l_j - l_i = d$) then $l_j < d$ (resp. $l_j > d$) and there is no $l_{i'} \in L$ such that $l_j - l_{i'} = d$ (resp. $l_j + l_{i'} = d$). This observation implies that every occurrence of d in S_+ induces a token in $S_0 \cap S_d$ which is different from all the tokens induced by occurrences of d in S_-. Then, the first part of the proof ($f_d \leq |S_0 \cap S_d|$) is completed.

Secondly, we prove that $f_d \geq |S_0 \cap S_d|$, that is, for every token z_x in $S_0 \cap S_d$ there is an occurrence of d in $S_+ \cup S_-$. Clearly $x = l_j \in L$.

- If $x > d$ then, there is $l_i \in L$ such that $x - l_i = d$. Also, if there is a token $z_{x'} \in S_0 \cap S_d$ such that $z_{x'} \neq z_x$ and $x' > d$ then, there is a value $l_{i'} \in L$, $l_{i'} \neq l_i$ such that $x' - l_{i'} = d$. Therefore, every token $z_x \in S_0 \cap S_d$ with $x > d$ induces a different occurrence of d in S_-.

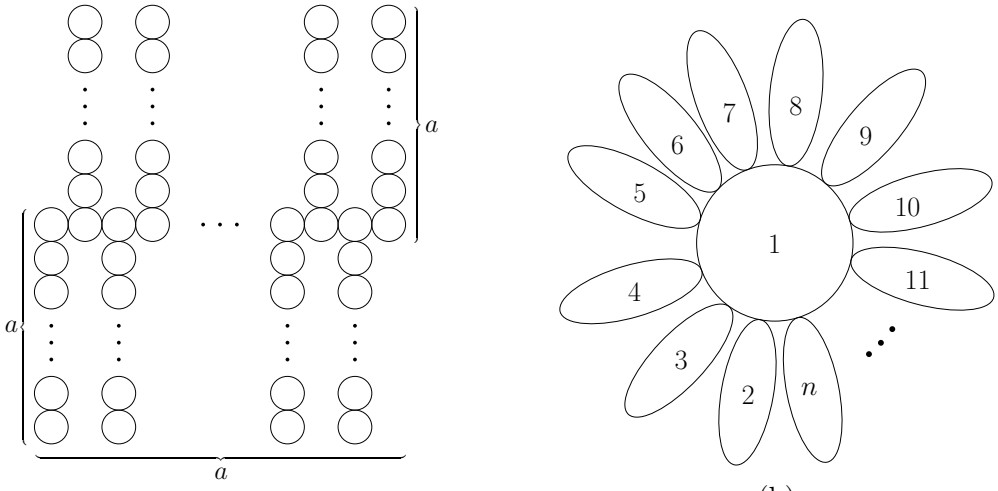
- If $x < d$ then, there is $l_i \in L$ such that $((d + l_i) \mod n) = x$. Taking into account that $0 < x < d < n$ and $l_i < n$, we get that $n < d + l_i < 2n$, then $d - (n - l_i) = x$. By rewriting the last equation as $d = x + (n - l_i)$ and taking into account that $(n - l_i) \in L$ (by properties of circulant graphs), an occurrence of d in S_+ is induced. Analogously to the previous case, if there is another token $z_{x'} \in S_0 \cap S_d$ such that $z_{x'} \neq z_x$ and $x' < d$ then, there is a value $l_{i'} \in L$, $l_{i'} \neq l_i$ such that $d - (n - l_{i'}) = x'$. Therefore, every token $z_x \in S_0 \cap S_d$ with $x < d$ induces a different occurrence of d in S_+.

The proof that $|S_0 \cap S_d| = f_d$ is now completed and, by using the definitions of mode and 2-isolation resilience we conclude that d is the mode of $S_+ \cup S_-$. □

From the previous lemma the following theorem is deduced:

Theorem 3.4.30. *The 2-isolation resilience of a SCS whose communication graph is a tree can be computed in $O(t^2 + n)$ time where, n is the number of trajectories in the system and t is the number of different tie lengths.*

Proof. In a preprocessing step, find the set of tie lengths $L = \{l_1, ..., l_t\}$ by traveling along the single ring of the system. This step takes $O(n)$ time. After that, by computing $l_j + l_i$ and $l_j - l_i$ for all $(l_i, l_j) \in L \times L$ we can find the mode d (and its frequency f) in the multiset $S_+ \cup S_-$ of Lemma 3.4.29. This step takes $O(t^2)$ time. Finally, the 2-isolation resilience is $2t - f$. □



(a) (b)

Figure 3.26: Examples of lower bounds in the number of tie lengths of a SCS. (a) A tree where $n = a^2$. The lengths of the ties are $L = \{1, 2, \ldots, a, 2a, 3a, \ldots, a^2 - a, a^2 - (a-1), \ldots, a^2 - 1\}$ and $|L| = 3a - 3$. (b) A star tree with non-circular trajectories where n could be arbitrarily large and $L = \{1, n-1\}$.

The following remark shows cases in which the application of the previous theorem conducts to a very good performance ini the computation of the 2-isolation resilience.

Remark 3.4.31. *If t is $O(\sqrt{n})$ then, computing the 2-isolation resilience takes linear time (Theorem 3.4.30). For example, consider a SCS with $n = a^2$ trajectories connected between them as it is shown in Figure 3.26a. It is easy to check that $t = 3a - 3$, then $t \in O(\sqrt{n})$. Also, considering non-circular and degenerated trajectories we can build a SCS whose communication graph is a star tree with $t = 2$ for any arbitrary large value of n, see Figure 3.26b. In these cases, the algorithm described in the proof of Theorem 3.4.30 has very good performance (linear time).*

Not everything is good news, there are some bad cases in which the idea of Theorem 3.4.30 is not so good, see the following remark.

Remark 3.4.32. *If t is $\Omega(n)$ (e.g., the communication graph of the system is a simple path), the idea of Theorem 3.4.30 takes $O(n^2)$ time.*

In the following, we show another idea to improve the time complexity of Theorem 3.4.30 when $t \in \Omega(n)$.

Lemma 3.4.33. *Let $L = \{l_1, \ldots, l_t\}$ be the set of tie lengths in the single ring of a SCS whose communication graph is a tree. Let $d \in \mathbb{N}$, $0 < d < n$. Let $p(x) = x^{l_1} + x^{l_2} + \cdots + x^{l_t}$. The coefficients of x^d and x^{n+d} in $p^2(x)$ are the number*

of occurrences of d in the multisets

$$S_+ = \{l_j + l_i \mid (l_i, l_j) \in L \times L,\ l_j + l_i \notin L,\ l_j + l_i < n\} \text{ and}$$
$$S_- = \{l_j - l_i \mid (l_i, l_j) \in L \times L,\ l_j - l_i \notin L,\ l_j - l_i > 0\}, \text{ respectively.}$$

Proof. Observe that

$$\begin{array}{rccccccccc} p^2(x) = & x^{l_1+l_1} & + & x^{l_1+l_2} & + & \dots & + & x^{l_1+l_t} & + \\ & x^{l_2+l_1} & + & x^{l_2+l_2} & + & \dots & + & x^{l_2+l_t} & + \\ & \vdots & & \vdots & & & & \vdots & \\ & x^{l_t+l_1} & + & x^{l_t+l_2} & + & \dots & + & x^{l_t+l_t}. & \end{array}$$

For every pair $(l_i, l_j) \in L \times L$ we have that:

- If $l_i + l_j = d$ then, an occurrence of d is induced in S_+ and a term x^d appears in $p^2(x)$. Therefore, the coefficient of x^d in $p^2(x)$ is the number of occurrences of d in S_+.
- If $l_i + l_j = n + d$, a term x^{n+d} appears in $p^2(x)$. Also, we have that $(n - l_i) \in L$ (by properties of circulant graphs), then $((n - l_i), l_j) \in L \times L$ and $l_j - (n - l_i) = d$. Therefore, an occurrence of d is induced in S_- and the coefficient of x^{n+d} in $p^2(x)$ is the number of occurrences of d in S_-.

□

Notice that the degree of $p(x)$ is less than n (recall that $\max\{l_1, \dots, l_t\} \leq n-1$), then its coefficient representation is a list of $O(n)$ elements. Moreover, the degree of $p^2(x)$ is less than $2n - 1$ and, its coefficient representation uses $O(n)$ elements as well. Having this representation of $p^2(x)$, we can find the mode of $S_+ \cup S_-$ and its frequency in linear time. The straightforward method to compute $p^2(x)$ from $p(x)$ takes $O(n^2)$ time, but, using the fast Fourier transform, or FFT, it can be done in $O(n \log n)$ time [35] (Chapter 30 - Polynomials and the FFT). These ideas support the following result:

Theorem 3.4.34. *The 2-isolation resilience of a SCS whose communication graph is a tree can be computed in $O(n \log n)$ time where, n is the number of trajectories in the system.*

Proof. In a preprocessing step, find the set of tie lengths $L = \{l_1, ..., l_t\}$ by traveling along single ring of the system. This step takes $O(n)$ time. After that, make a coefficient representation of $p(x) = x^{l_t} + \cdots + x^{l_1}$, it takes $O(n)$ time. Then, by using the FFT, compute $p^2(x)$ in $O(n \log n)$ time.

Let $c(l)$ denote the coefficient of x^l in $p^2(x)$. Find the value l^* such that:

$$c(l^*) + c(l^* + n) = \max_{1 \leq l < n, l \notin L} \{c(l) + c(n + l)\}.$$

This last step takes $O(n)$ time. The 2-isolation resilience is $2t - c(l^*) - c(l^* + n)$. □

The following corollary concludes our analysis of the problem of computing the 2-isolation resilience of a SCS whose communication graph is a tree. It is straightforward deduced from theorems 3.4.30 and 3.4.34.

Corollary 3.4.35. *The value of the 2-isolation resilience in a SCS whose communication graph is a tree can be computed in $O(n \log n)$ time in the worst case. Moreover, if the number of tie lengths is $O(\sqrt{n})$, then, the 2-isolation resilience can be computed in linear time.*

In the following we focus on computing the k-isolation resilience of a SCS whose communication graph is a tree.

Let $L = \{l_1, \ldots, l_t\}$ be the set of tie lengths in the system. Let $Z = \{0, \ldots, n-1\}$ be a circular numbering of the tokens such that there is a correspondence between two tokens i and j if and only if $((j-i) \mod n) \in L$. For all $S \subset Z$, let $N(S) = \{((l+i) \mod n) \mid l \in L, i \in S\}$ denote the set of neighboring tokens of S. To compute the k-isolation resilience of a SCS whose communication graph is a tree, we propose the following procedure:

With out loss of generality, set 0 as the first isolated token. Then, for every $(k-1)$-set S of isolated tokens such that $0 \in S$: Let $A_S = Z \setminus (S \cup N(S))$ be the set of available tokens. Now, compute the mode m of multiset $S_+ \cup S_-$ where

$$S_+ = \{z + l \mid (z,l) \in N(S) \times L,\ z + l < n,\ z + l \in A_S\} \text{ and}$$
$$S_- = \{z - l \mid (z,l) \in N(S) \times L,\ z - l > 0,\ z - l \in A_S\}.$$

Compute $\rho(S) = |N(S)| + t - f$ where f is the frequency of m in $S_+ \cup S_-$. Compute the k-isolation resilience as minimum of $\rho(S)$ over all possible sets S.

The correctness of this procedure can be proven using the same ideas of Lemma 3.4.29. Theorems 3.4.36 and 3.4.37 focus on the time complexity of this procedure by using two different approach to compute the mode m of multiset $S_+ \cup S_-$.

Theorem 3.4.36. *In a SCS whose communication graph is a tree, the k-isolation resilience, $k \geq 3$, can be computed in $O(n^{k-2}t \cdot \min(n, kt))$ time.*

Proof. In a preproccessing L can be computed in linear time. Note that, the number of $(k-1)$-sets of isolated tokens that contains 0, is $O(n^{k-2})$. Using the same ideas of Theorem 3.4.24 to compute these sets takes $O(ktn^{k-2})$ time. By other hand, for every computed set S we have that $|N(S)| \leq \min\{n, (k-1)t\}$. Therefore, a straightforward approach to compute the mode m of multiset $S_+ \cup S_-$ takes $O(t \cdot \min(n, kt))$ time. Taking into account that we need to compute m for all computed $(k-1)$-sets we have total computing time $O(ktn^{k-2} + n^{k-2}t \cdot \min(n, kt)) = O(n^{k-2}t \cdot \min(n, kt))$ time. □

Theorem 3.4.37. *In a SCS whose communication graph is a tree, the k-isolation resilience, $k \geq 3$, can be computed in $O(n^{k-2}(tk + n \log n))$ time.*

Proof. The only difference with the previous theorem is that, in here, we compute the mode m of multiset $S_+ \cup S_-$ by applying polynomial multiplication (using FFT). Let the polynomials $p(x) = \sum_{l \in L} x^l$ and $q(x) = \sum_{z \in N(S)} x^z$. Notice that degree of both polynomials is bounded by n, then, their multiplication takes $O(n \log n)$ time. Let $c(l)$ denote the coefficient of x^l in $(p \cdot q)(x)$. Find the value l^* such that:

$$c(l^*) + c(l^* + n) = \max_{1 \leq l < n, l \in A_S} \{c(l) + c(n + l)\}.$$

This last step takes $O(n)$ time. The mode of $S_+ \cup S_-$ is l^* and its frequency is $c(l^*) + c(l^* + n)$.

Finally, taking into account that computing all possible $(k-1)$-sets of isolated tokens that contains 0 takes $O(ktn^{k-2})$ time, the result follows. □

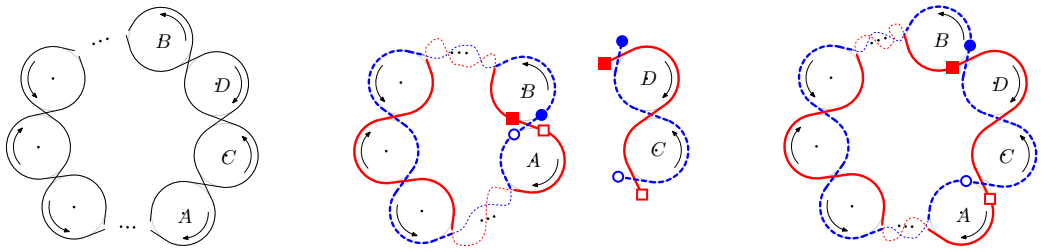
Remark 3.4.38. *For small values of k: if $t \in O(\sqrt{n})$ then, it is better to use Theorem 3.4.36 because the time complexity becomes $O(kn^{k-1})$. By other hand, if $t \in \Omega(n)$ then, it is better to use Theorem 3.4.37 because the time complexity becomes $O(n^{k-1}(k + \log n)) = O(n^{k-1} \log n)$.*

Cycles

Lemma 3.4.39. *Let G be the communication graph of a m-partial SCS. If G is a cycle, then the system has exactly two rings, one with CW direction and the other with CCW direction. Furthermore, every edge of G corresponds to a crossing of the two rings.*

Proof. We proceed by induction on the number $2M$ of trajectories (nodes) in G (recall that G is bipartite, so every cycle has even length). If $M = 2$, we have 4 trajectories, and the claim holds, as shown in Figure 3.8a. Assume as inductive hypothesis that, for a fixed value M, the claim holds for every cycle graph with $2M$ trajectories. Now, let us consider a cycle graph G with $2(M + 1)$ trajectories (Figure 3.27a). If we remove two consecutive trajectories (C and D) and "glue" the ends in the broken section we obtain a cycle graph G' with $2M$ trajectories (Figure 3.27b). By the inductive hypothesis, the claim holds for G'. Figure 3.27b shows the clockwise ring using solid stroke in red, and the other one using dashed stroke in blue. The sections of rings in the removed two circles are also shown in Figure 3.27b. After that, inserting again the two removed trajectories into the original position we obtain a cycle graph with $2(M+1)$ trajectories, reestablishing the claim, see Figure 3.27c. Note that, as depicted in Figure 3.27c, when we reinsert the removed trajectories, we match terminals (solid-circle to solid-circle, solid-square to solid-square, etc.). □

Lemma 3.4.40. *In an m-partial SCS, whose communication graph is a cycle with rings r and r', a robot in r is isolated if and only if r' is empty of robots (tokens).*



(a) (b) (c)

Figure 3.27: Induction proof of Lemma 3.4.39.

Proof. Suppose that there are tokens z and z' in r and r', respectively. From Property 3.2.5 and Lemma 3.4.39 we deduce that z and z' will meet each other, so, there is a correspondence between them. Also, note that there is no correspondence between tokens of the same ring. Then, from Lemma 3.4.1 it follows that a robot in r bearing a token z is starving if and only if the ring r' is empty of tokens. □

Using Lemma 3.4.40 and Lemma 3.2.13 we conclude:

Theorem 3.4.41. *Consider a system whose communication graph is a cycle and let r and r' be the two rings with lengths $2\pi l$ and $2\pi l'$, respectively. The isolation number of the system is $\max\{l, l'\}$ and the k-isolation resilience is $\min\{l, l'\}$ if $k \leq \max\{l, l'\}$ and infinity, otherwise.*

Remark 3.4.42. *If the communication graph is a cycle with isolation number ζ, then for all $k \leq \zeta$, the value of the k-isolation resilience matches the value of the coverage resilience of the system (Theorem 3.4.41 and Corollary 3.3.3).*

To close this subsection, notice that to compute the k-isolation resilience of an SCS whose communication graph is a cycle we can use the same linear time algorithm described in Theorem 3.3.4.

Grids

In this subsection we revisit the grid SCS introduced in subsection 3.3.1. We show a set of results that allow us to calculate the k-isolation resilience with a closed formula. All the results presented in previous sections are directly based on the properties of rings (their lengths, number, or topology). However, the following results, surprisingly, do not use these tools.

Consider the movement lattice introduced in subsection 3.3.1. Analyzing the movement of a token in the movement lattice of a grid SCS, we arrive at the following remark:

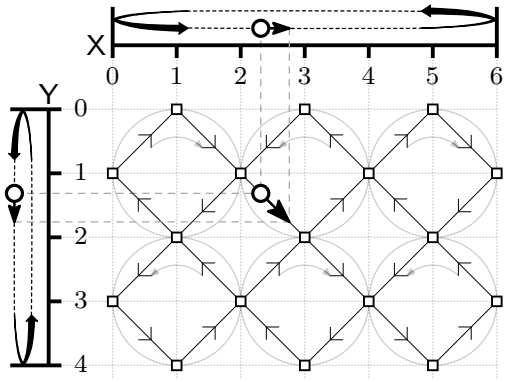


Figure 3.28: The small hollow disk represents a token moving in the movement lattice. The horizontal and vertical projections of the token motion are also shown.

Remark 3.4.43. *In the lattice of a grid SCS, every token moves with constant speed according to a* mathematical billiard *pattern of motion [130]. Moreover, the* X-motion *(resp.* Y-motion*) of a token, which is the projection of the token motion onto the X-axis (resp. Y-axis), is a periodic movement of a ball on a line segment that bounces off the ends, see Figure 3.28. Notice also that the speed in the X-axis (resp. Y-axis), hence on the lattice as well, of all the tokens is the same.*

Lemma 3.4.44. *In a partial grid SCS, if two robots occupy trajectories in the same row (resp. column) at some instant of time, then their respective tokens have always been and will always be in the same row (resp. column) of the movement lattice.*

Proof. Let u and v be robots in the same row, say row i, at some time t (the case of robots in the same column is treated similarly). Suppose that robot u is at position α. Since the system is synchronized, the position of robot v is at α or $\pi - \alpha$, see Figure 3.29. However, independent of whether the position of v is α or $\pi - \alpha$, they both have the same direction along the Y axis (both going down or both going up) due to synchronization. Then, due to the periodic constant motion of the tokens in Y axis (Remark 3.4.43), the tokens of u and v are in the same row at any time (before t and after t). The lemma follows. □

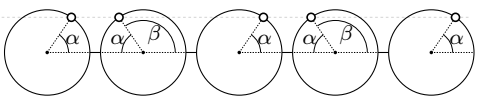


Figure 3.29: The white points represent the positions of robots in trajectories of a same row in a grid SCS, $\beta = \pi - \alpha$.

Theorem 3.4.45. *A robot (token) in a partial grid SCS is isolated if and only if there is no other robot (token) in the same row or column.*

Proof. ($\Rightarrow$) Let u be an isolated robot. For the sake of contradiction, suppose that there is another robot v in the same row at some time t (the case when they are in the same column is analyzed similarly). Let z and z' be the tokens of u and v at time t, respectively. Since the X-motion of each of these tokens is periodic and has the same constant speed, it is easy to see that after some time, say $\triangle t$ units of time, z and z' will meet each other in the X-axis projection. Thus, the tokens z and z' have the same x and y coordinates in the movement lattice at time $t + \triangle t$. Since z and z' at time $t + \triangle t$ belong to two different robots (one robot is u and the other robot is not necessarily v), these robots are at the communication link between two neighboring trajectories. Therefore, robot u is not isolated, a contradiction.

($\Leftarrow$) It suffices to show that, if a robot u is not isolated, then there is another robot in the same row or column. Robot u will be communicating with another robot, say v, at some time t. There are two possibilities for the communication link in the grid. The trajectories of u and v at time t are either in the same row or in the same column. Either way, the theorem follows. □

Finally, by induction on k and the pigeonhole principle, we conclude:

Theorem 3.4.46. *The isolation number in an $M \times N$ grid SCS is $\min(M, N)$ and its k-isolation resilience is $k(M + N - 2) - k(k - 1)$ if $k \leq \min(M, N)$ and infinity, otherwise.*

3.5 Computing broadcasting resilience

The robots of a SCS conform a *connected network* because every pair of robots can send/receive messages between each other, possibly by multiple relays between neighboring robots.

We can model the message transmission by means of the following protocol: the messages are carried by the tokens instead of the robots. If a token is carrying a message and meets another token who does not know it, then, after the meeting, both of them know the message. Therefore, when a robot u sends a message at time t, we assume that u puts the message into its token at time t. At time $t' > t$, we say that a robot u' knows the message if and only if it is bearing a token that has the message.

Lemma 3.5.1. *Let $\mathcal{F}$ be an m-partial SCS. If at some time t robot u of $\mathcal{F}$ is bearing a token z then u will bear z periodically. More formally, if u in $\mathcal{F}$ is bearing token z at time t, then there exist a value $\sigma \in \mathbb{N}$ ($\sigma > 0$) such that u is bearing z at every time of the form $t + \sigma k$ with $k \in \mathbb{N}$.*

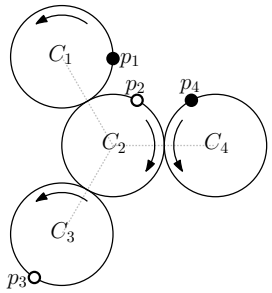


Figure 3.30: A 2-partial SCS at time t_0, the solid points are the positions occupied by the surviving robots and the hollow points are the positions where a removed robot should be.

Proof. Let $U = \{u_{i_1}, \ldots, u_{i_m}\}$ be the set of live robots in $\mathcal{F}$ and let Z be the set of tokens in $\mathcal{F}$. Fix a time t_0. For every trajectory C_j $(1 \leq j \leq n)$ in the system, let p_j be the position of C_j where there is or there should be (due to the synchronization of the system) a robot at time t_0, see Figure 3.30. Let $P = \{p_1, \ldots, p_n\}$. Notice that if a robot u is at a point $p \in P$ then after one time unit u is at a point p' that is in P as well. For every $l \in \mathbb{N}$ and $1 \leq j \leq m$, let $\left(\rho_j^{(l)}, \tau_j^{(l)}\right)$ be an ordered pair where $\rho_j^{(l)}$ and $\tau_j^{(l)}$ are the position and token of robot u_{i_j} at time $t_0 + l$, respectively. Notice that $\rho_j^{(l)} \in P$ and $\tau_j^{(l)} \in Z$ for all j, l.

Let $X(l) = \left(\left(\rho_1^{(l)}, \tau_1^{(l)}\right), \ldots, \left(\rho_m^{(l)}, \tau_m^{(l)}\right)\right)$. Note that $X(l)$ describes the state of the system at time $t_0 + l$, a kind of snapshot of the system at time $t_0 + l$. Let us analyze the infinite sequence $\mathcal{X} = X(0), X(1), X(2), \ldots$. The values of this sequence are the columns of Table 3.1. Now, recall that $\left(\rho_j^{(l)}, \tau_j^{(l)}\right)$ is in $P \times Z$ for all l, j, then $X(l)$ is in $(P \times Z)^m$ for all l. Notice that $|P| = n$ and $|Z| = m$, thus $|(P \times Z)^m| = (nm)^m$. It follows, using the pigeonhole principle, that the sequence $\mathcal{X}$ repeats values. Let σ be a natural number such that $X(\sigma)$ is the first repeated value in $\mathcal{X}$. That is, there is a value $l' \in \mathbb{N}$, $0 \leq l' < \sigma$ such that $X(l') = X(\sigma)$, see Table 3.1.

We claim that $l' = 0$. Suppose that $l' > 0$. As a consequence of the behavior of a partial SCS it is easy to show that $X(l'+1) = X(\sigma+1)$ and $X(l'-1) = X(\sigma-1)$. Then the value $X(\sigma-1)$ is the first repeated value in $\mathcal{X}$, a contradiction! Therefore $l' = 0$ and σ is the period of the system, i.e., $X(l) = X(l + \sigma)$ for all $l \in \mathbb{N}$. □

Lemma 3.5.2. *Let $\mathcal{F}$ be an m-partial SCS. Let $\mathcal{T}_\mathcal{F}$ be the meeting graph of $\mathcal{F}$. If at time t a robot u is bearing a token z then at time $t' > t$ robot u is bearing a token z' which is in the same connected component of z in $\mathcal{T}_\mathcal{F}$.*

Proof. From the nature of the meeting graph, we know that if a robot u with token z exchanges its token with a robot u' with token z', then z and z' are adjacent in

	0	1	$\cdots$	σ	$\cdots$
u_{i_1}	$\left(\rho_1^{(0)}, \tau_1^{(0)}\right)$	$\left(\rho_1^{(1)}, \tau_1^{(1)}\right)$	$\cdots$	$\left(\rho_1^{(\sigma)}, \tau_1^{(\sigma)}\right)$	$\cdots$
u_{i_2}	$\left(\rho_2^{(0)}, \tau_2^{(0)}\right)$	$\left(\rho_2^{(1)}, \tau_2^{(1)}\right)$	$\cdots$	$\left(\rho_2^{(\sigma)}, \tau_2^{(\sigma)}\right)$	$\cdots$
$\vdots$	$\vdots$	$\vdots$		$\vdots$	
u_{i_m}	$\left(\rho_m^{(0)}, \tau_m^{(0)}\right)$	$\left(\rho_m^{(1)}, \tau_m^{(1)}\right)$	$\cdots$	$\left(\rho_m^{(\sigma)}, \tau_m^{(\sigma)}\right)$	$\cdots$

Table 3.1: The j-th row represents the sequence of transitions of the robot u_{i_j} at times $t_0, t_0+1, t_0+2, \ldots$.

the meeting graph. Then, the sequence of token's exchanges made by u from time t to time t' defines a path in $\mathcal{T}_{\mathcal{F}}$. □

Lemma 3.5.3. *Let $\mathcal{F}$ be an m-partial SCS. Let $\mathcal{T}_{\mathcal{F}}$ be the meeting graph of $\mathcal{F}$. Let u and u' be robots in $\mathcal{F}$ bearing tokens z and z' at time t, respectively. A message sent by u at time t is delivered to u' if and only if z and z' are in the same connected component of $\mathcal{T}_{\mathcal{F}}$.*

Proof. ($\Rightarrow$) Suppose that a message sent by u at time t (hence, associated with z) is delivered to u' at time t'. Let τ be the token held by u' at time t'. It is easy to see that if τ contains the message then there is a path in $\mathcal{T}_{\mathcal{F}}$ between z and τ. Thus, from Lemma 3.5.2, we deduce that z and z' are in the same connected component of $\mathcal{T}_{\mathcal{F}}$.

($\Leftarrow$) Suppose that z and z' are in the same connected component of $\mathcal{T}_{\mathcal{F}}$. Thus, the message in z will be delivered to z' at some time. From Lemma 3.5.1, we deduce that the message will reach u'. □

Remark 3.5.4. *From the previous lemma we deduce that if it is possible to send a message at time t from robot u to robot u', then it is also possible to send a message at time t from u' to u.*

From the previous remark and Lemma 3.5.3 we arrive to the following corollary:

Corollary 3.5.5. *Let $\mathcal{F}$ be an m-partial SCS. Let $\mathcal{T}_{\mathcal{F}}$ be the meeting graph of $\mathcal{F}$. There is a loss of connectivity in $\mathcal{F}$ if and only if $\mathcal{T}_{\mathcal{F}}$ is disconnected or if it consists of a single vertex.*

Let $G = (V, E)$ be a connected graph. A *vertex-separator* is a set $S \subset V$ such that the subgraph induced by $V \setminus S$ is disconnected. The *vertex-connectivity* of G, denoted by $\kappa(G)$ is $|V| - 1$ if $G = K_n$ (the complete graph); otherwise, $\kappa(G)$ is the cardinality of a smallest vertex-separator.

From Lemma 3.2.20, Corollary 3.5.5 and by using the previous definition of connectivity of a graph, the next result follows:

Theorem 3.5.6. *Let $\mathcal{F}$ be a SCS and let $\mathcal{T}_{\mathcal{F}}$ be the meeting graph of $\mathcal{F}$. The broadcasting resilience of $\mathcal{F}$ is $\kappa(\mathcal{T}_{\mathcal{F}})$. And, it can be computed in $O(n\kappa(\mathcal{T}_{\mathcal{F}}) \cdot \min\{\kappa^3(\mathcal{T}_{\mathcal{F}}) + n, n\kappa(\mathcal{T}_{\mathcal{F}})\})$ time.*

Proof. The graph $\mathcal{T}_{\mathcal{F}}$ can be computed in $O(n^2)$ time (Lemma 3.4.23). Then, the vertex-connectivity of $\mathcal{T}_{\mathcal{F}}$ can be computed in $O(n\kappa(\mathcal{T}_{\mathcal{F}}) \cdot \min\{\kappa^3(\mathcal{T}_{\mathcal{F}}) + n, n\kappa(\mathcal{T}_{\mathcal{F}})\})$ time [63] and, the result follows. □

By using the definitions of the resilience measures we have:

Corollary 3.5.7. *The broadcasting resilience of a SCS $\mathcal{F}$ is less than or equal to its 1-isolation resilience.*

In Section 3.4.3, it is shown that the 1-isolation resilience of a SCS can be computed in $\tilde{O}(n)$, therefore, we can compute a bound for the broadcasting resilience of a SCS in almost linear time. The next subsections focuses on improving the time complexity of Theorem 3.5.6 on some configurations that appear in practical applications: trees, cycles and grids.

3.5.1 Broadcasting resilience for trees, cycles and grids

In this section we are going to focus in computing the broadcasting resilience for some specific cases of the communication graph: trees, cycles and grids.

Trees

Let $\mathcal{F}$ be a SCS whose communication graph is a tree. From Lemma 3.4.12 we know that the meeting graph $\mathcal{T}_{\mathcal{F}}$ of $\mathcal{F}$ is a circulant graph. Using Remark 3.4.11 we have that $\mathcal{T}_{\mathcal{F}} = C_n J$ where n is the number of trajectories in the system and J is the set of jumps. Notice that the number of edges of $\mathcal{T}_{\mathcal{F}}$ could be quadratic (i.e., if the trajectories are disposed in a path then, the meeting graph is a complete graph). So, computing explicitly every edge of $\mathcal{T}_{\mathcal{F}}$ takes $O(n^2)$ time in the worst case. However, notice that $|J| \leq \left\lfloor \frac{n}{2} \right\rfloor$ and computing J takes linear time (see Lemma 3.4.27).

Lemma 3.5.8. *Let $\mathcal{F}$ be a SCS with a single ring. The broadcasting resilience of $\mathcal{F}$ can be computed in $O(n^{3/2})$ time.*

Proof. Computing the set J of jumps of the circulant meeting graph takes linear time (Lemma 3.4.27). Then, the connectivity of $C_n J$ can be computed in $O(n^{3/2})$ time using the algorithm proposed in [20, 88]. By Theorem 3.5.6, the result follows. □

From the lemma above and Lemma 3.3.6, the following result is directly deduced:

Corollary 3.5.9. *The broadcasting resilience of a SCS whose communication graph is a tree can be computed in $O(n^{3/2})$ time.*

Cycles

Lemma 3.5.10. *Let $\mathcal{F}$ be a SCS whose communication graph is a cycle. The broadcasting resilience of $\mathcal{F}$ is* $\min\{l, l'\}$ *where $2l\pi$ and $2l'\pi$ are the lengths of the two rings in $\mathcal{F}$. Moreover, the broadcasting resilience can be computed in linear time.*

Proof. From Property 3.2.5, Remark 3.2.6 and Lemma 3.4.39 it follows that the meeting graph of $\mathcal{F}$ is the complete bipartite graph $K_{l,l'}$ and therefore the minimum separator vertex set of $K_{l,l'}$ has cardinality $\min\{l, l'\}$. The linear time algorithm described in Theorem 3.3.4 allows us to compute the rings of the system and their lengths. □

Corollary 3.5.11. *Let $\mathcal{F}$ be a SCS whose communication graph is a cycle. Let $2l\pi$ and $2l'\pi$ be the lengths of the two rings in $\mathcal{F}$. The three measures, broadcasting resilience, coverage resilience and k-isolation resilience (for all $k \leq \max\{l, l'\}$), have the same value, which is* $\min\{l, l'\}$ *and, this value can be computed in linear time.*

Grids

Lemma 3.5.12. *Let $\mathcal{F}$ be an $N \times M$ SCS. The broadcasting resilience of $\mathcal{F}$ is $N + M - 2$.*

Proof. From Theorem 3.4.45 we deduce that the meeting graph of $\mathcal{F}$ is a complete graph if and only if $N = 1$ or $M = 1$. In any of these cases the lemma holds.

If $N \geq 2$ and $M \geq 2$, the meeting graph $\mathcal{T}_{\mathcal{F}}$ has a separator set $S \subset V(\mathcal{T}_{\mathcal{F}})$ such that $|S| = \kappa(\mathcal{T}_{\mathcal{F}})$. Now, from Theorem 3.4.46 we know that the 1-isolation resilience of $\mathcal{F}$ is $N + M - 2$, then, by using Corollary 3.5.7, we have that $\kappa(\mathcal{T}_{\mathcal{F}}) \leq N + M - 2$. Then:

$$|V(\mathcal{T}_{\mathcal{F}}) \setminus S| = N \cdot M - \kappa(\mathcal{T}_{\mathcal{F}}) \geq N \cdot M - N - M + 2. \tag{3.5}$$

In the following we prove that:

$$|V(\mathcal{T}_{\mathcal{F}}) \setminus S| \leq N \cdot M - N - M + 2. \tag{3.6}$$

Let $A \neq \emptyset$ and $B \neq \emptyset$ form a bipartition of $V(\mathcal{T}_{\mathcal{F}}) \setminus S$ such that $\{a, b\} \notin E(\mathcal{T}_{\mathcal{F}})$ for all $a \in A$ and $b \in B$. Let $\texttt{rows}(A) \subset \{1...N\}$ and $\texttt{rows}(B) \subset \{1...N\}$ denote the sets of rows occupied by the tokens in A and B, respectively. Analogously, let $\texttt{cols}(A) \subset \{1..M\}$ and $\texttt{cols}(B) \subset \{1..M\}$ denote the sets of columns occupied by the tokens in A and B, respectively. Notice that $\texttt{rows}(A) \cap \texttt{rows}(B) = \texttt{cols}(A) \cap \texttt{cols}(B) = \emptyset$.

Take $r \in \texttt{rows}(A)$ and $c \in \texttt{cols}(A)$, let z be the token of $\mathcal{F}$ on the circle of row r and column c. If z is not in A then z is in S. Let $A' = A \cup \{z\}$

and $S' = S \setminus \{z\}$. Notice that A' and B form a bipartition of $V(\mathcal{T}_{\mathcal{F}}) \setminus S'$ and there is no edge from A' to B. Therefore, S' is a separator set and $|S'| < |S|$ which is a contradiction. Thus, for every $r \in \texttt{rows}(A)$ and $c \in \texttt{cols}(A)$ the token on the circle (r, c) is in A and $|A| = |\texttt{rows}(A)| \cdot |\texttt{cols}(A)|$. Analogously, we can prove that $|B| = |\texttt{rows}(B)| \cdot |\texttt{cols}(B)|$. To simplify the notation, let $|\texttt{rows}(A)| = r_A$, $|\texttt{cols}(A)| = c_A$, $|\texttt{rows}(B)| = r_B$ and $|\texttt{cols}(B)| = c_B$. Then: $|V(\mathcal{T}_{\mathcal{F}}) \setminus S| = |A| + |B| = r_A \cdot c_A + r_B \cdot c_B$.

Since $r_B \leq N - r_A$ and $c_B \leq M - c_A$, then:

$$r_A \cdot c_A + r_B \cdot c_B - N \cdot M + N + M - 2 \leq r_A \cdot c_A + (N - r_A) \cdot (M - c_A) - N \cdot M + N + M - 2.$$

The right part of the above inequality can be rewritten as:

$$(1 - c_A)(N - r_A - 1) + (1 - r_A)(M - c_A - 1).$$

Notice that $(1 - c_A) \leq 0$ and $(1 - r_A) \leq 0$. On the other hand, $(N - r_A - 1) \geq 0$ and $(M - c_A - 1) \geq 0$. This proves inequality (3.6) and the stated lemma follows from inequalities (3.5) and (3.6).

□

3.6 Conclusions

Area coverage in cooperative robot networks is a fundamental component of many applications. For instance, a group of UAVs can form a network and accomplish complicated missions such as rescue, searching, patrolling and mapping. There are two main issues which must be considered when developing a solution with a cooperative robot network: coverage and communication.

In this chapter we propose various quality measures of a synchronized system of robots related to the robustness of the network in the presence of failures. Computing these measures leads to interesting combinatorial problems from the theoretical point of view. With respect to communication between robots, we considered two quality measures: the k-isolation resilience and the broadcasting resilience. The k-isolation resilience is the minimum number of robots whose removal may cause the isolation of at least k surviving robots. Although the problem of computing this measure is NP-hard in general, even for trees, we showed how to solve the problem when the system is small or for small values of k. We also show how to compute the k-isolation resilience when the communication graph is a cycle (in linear time) or a grid (with an analytic expression) which are commonly used configurations in multi-UAVs applications. The broadcasting resilience is related to the capacity of the system for sending messages through the network, it can be computed in $O(n\kappa \cdot \min\{\kappa^3 + n, n\kappa\})$ time where n is the number of trajectories and κ is the broadcasting resilience value. Additionally, we showed how to improve this complexity on trees to $O(n^{3/2})$ time, on cycles to $O(n)$ time and, on grids to

constant time. For the study of the robustness with respect to area coverage, we introduced the concept of coverage resilience as the minimum number of robots whose removal may result in at least one non-covered trajectory segment. Notice that if we denote this measure as ρ_c, the removal of at most $\rho_c - 1$ robots from the system guarantees that the total area is covered. In order to bound the idle-time of the system, that is, the maximum time a point of the area is unattended by the robots, we also define the T-coverage resilience as the minimum number of robots we can remove so that the idle-time is at least T. We gave a linear time algorithm to compute these values of resilience for a general communication graph and we found closed form solutions depending on the input size for trees and grid-graphs. Moreover, we showed some relationship between the measures. For example, the three measures match when the communication graph is a cycle.

Notice that the time complexity of Theorem 3.5.6 depends on the algorithm exposed in [63] which computes the vertex-connectivity of any general graph. Then, it could be interesting to study if there is a better algorithm to compute the broadcasting resilience by using some specific properties of the vertex-connectivity in meeting graphs.

Chapter 4

Randomized Synchronized Communication Systems

STOCHASTIC multi-UAV systems have attracted considerable attention in the field of mobile robots. This approach brings several advantages, such as lower times to complete tasks, cost reduction, higher scalability, and more reliability, among others [150]. In a pure random mobility model, each node randomly selects its direction, speed, and time, independently of other nodes. Some models include random walks, random waypoints or random directions. See [26] for a comprehensive survey.

In this chapter we introduce some stochastic behavior for the surviving robots of a partial SCS (see Definition 3.1.1 of Chapter 2) in order to gain robustness in the system. Recall that in a partial SCS the surviving robots perform a deterministic protocol when they arrive to a link position: if there is a robot in the neighboring trajectory then they exchange information and remain in their respective trajectories, and, if there is no robot in the neighboring trajectory then the robot shifts to the neighboring trajectory. In this chapter we call **deterministic strategy** to this protocol. In Chapter 3 we have presented three problems that may appear in a partial SCS using the deterministic strategy: isolation (a robot always fails trying to meet a robot in the communication links), uncovering (some trajectory sections are non covered) and loss of connectivity (the system loses the capacity to relay messages between every pair of robots in the system). In this chapter, we introduce a random way to perform the shifting operations in order to avoid these drawbacks.

We are going to show that the random alternatives behave really well with very few robots in the system. Then, in scenarios where we have many subregions (trajectories) and a few robots its more convenient to use the random strategies

of this chapter.

4.1 Random strategies and measures

Two random strategies are proposed in this chapter as an alternative to the deterministic strategy proposed and studied in chapters 2 and 3. The first one is easier to study theoretically and the other one is more convenient in practical scenarios.

Random strategy: Every time a robot arrives at a communication link between its current trajectory and a neighboring one, it decides with probability $p = \frac{1}{2}$ if it remains in its trajectory or it passes to the neighboring one by a shifting operation. Notice that by following this protocol more than one robot may share the same position. For instance, if two robots arrive at a communication link and one of them decides to maintain its trajectory and the other one decides to make a shifting operation, then they will move together like a single robot. Working with aerial robots, this drawback can be solved by flying at different altitudes or by flying side by side within a safety margin. Also note that while the robots are flying together, they can maintain permanent communication.

Quasi-random strategy: Every time a robot arrives at a communication link, the following rule is applied: if there is no robot in the neighboring trajectory, then it decides with probability $p = \frac{1}{2}$ if it remains at its trajectory or if it passes to the neighboring one by a shifting operation. Otherwise, it remains in its trajectory. Note that with this protocol, no two robots travel on the same trajectory and a collision avoidance approach is not required in this case.

Definition 4.1.1. *We will call* randomized SCS (R-SCS) *to a synchronized system where the robots are applying the* ***random*** *strategy. A* quasi-randomized SCS (QR-SCS) *is defined analogously.*

Remark 4.1.2. *The behavior of a robot in a R-SCS is independent of the other robots; the movement of a robot in QR-SCS depends on the behavior of the other robots. Finally, the behavior of the robots following the deterministic strategy is totally co-dependent.*

In what follows we introduce three criteria to compare the performance of these strategies. The metrics carry valuable information regarding the coverage and the communication performance of a SCS.

- The **idle-time** is the average time that a point in the union of the trajectories remains unobserved by a robot.

- The **isolation-time** is the average time that a robot is without communication with any other robot.

- The **broadcast-time** is the average time elapsed from the moment a robot emits a message until it is received by all the other robots.

It is worth mentioning that these metrics are related to some resilience measures defined previously for SCS's: idle-time is related to coverage-resilience, defined the minimum number of robots whose removal may result in a non-covered subarea (see Definition 3.1.8), isolation-time is related to 1-isolation resilience, defined as the cardinality of a smallest set of robots whose failure suffices to cause that at least one surviving robots operates without communication (see Definition 3.1.6) and broadcast-time is related to broadcasting resilience, introduced as the minimum number of robots whose removal may disconnect the network (see Definition 3.1.7).

4.2 Theoretical results

In this section we present some theoretical bounds on the above metrics. We then compare these values with experimental results. First, we review some well-known notions of random walks. Random walks are the main tool that we will use to study the behavior of the metrics in a partial SCS. We follow the notation of [85]. Let $G = (V, E)$ be a connected (di)graph with n vertices. Starting at a vertex v_0, randomly select a neighbor v_1 of v_0. Then randomly select a neighbor of v_1 and so on. This process yields a sequence of vertices of G, which we call a *random walk* on G. Let v_t denote the vertex of the random walk at the t-th step. Let P_0 be the probability distribution from which v_0 was chosen, i.e., $P_0(v) = \Pr(v_0 = v)$ for all $v \in V$. We denote by $M = (p_{vw})_{v,w \in V}$ the transition matrix, i.e., p_{vw} is the probability of moving from vertex v to vertex w. Then, $P_t = M^t P_0$ denotes the probability distribution of v_t, i.e., $P_t(v) = \Pr(v_t = v)$ for all $v \in V$.

The *period* k of a vertex v is defined as $k = \gcd\{n > 0 | \Pr(v_t = v | v_0 = v) > 0\}$. In other words, v has period k if any subsequent visit to v occurs in multiples of k time steps. If $k = 1$, the vertex v is *aperiodic*. If every vertex in G is aperiodic, G is said to be *aperiodic*.

A probability distribution P that satisfies that $P = M \cdot P$ is called a *stationary distribution*. If the graph G is (strongly) connected, then the stationary distribution exists and it is unique. Moreover, if G is also aperiodic, P_t converges to the stationary distribution as $t \to \infty$.

4.2.1 The discrete model of a partial SCS

Let (α, δ) be the synchronization schedule of a partial SCS $\mathcal{F}$ with n trajectories $\{C_1, \ldots, C_n\}$. Consider the digraph $G = (V, E)$, where $V = \{1, \ldots, n\}$ and (i, j) is an arc of E from i to j if there is a path P (not necessarily in the same ring) of length 2π from $\alpha_i = \alpha(C_i)$ to $\alpha_j = \alpha(C_j)$ following the assigned travel directions in the circles. See Figure 4.1, where a SCS is shown and a path between

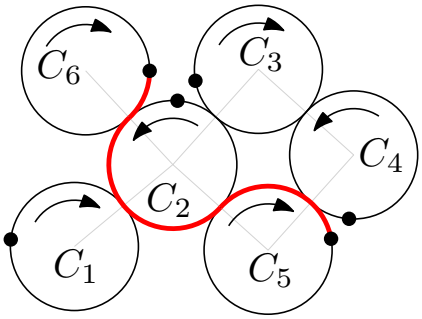


Figure 4.1: The solid points are the starting positions of synchronization schedule. The path (in red stroke) from α_6 to α_5 has length 2π and follows the travel directions along the visited trajectories. This path traverses four communication links.

	1	2	3	4	5	6
1	1	1	1	1	1	0
2	1	1	1	1	1	1
3	0	1	1	1	1	0
4	0	1	1	1	1	0
5	0	1	1	1	1	0
6	1	1	1	1	1	1

(a)

	1	2	3	4	5	6
1	$\frac{1}{2}$	$\frac{1}{8}$	$\frac{1}{8}$	$\frac{1}{8}$	$\frac{1}{8}$	0
2	$\frac{1}{4}$	$\frac{1}{16}$	$\frac{1}{16}$	$\frac{1}{16}$	$\frac{1}{16}$	$\frac{1}{2}$
3	0	$\frac{1}{4}$	$\frac{1}{4}$	$\frac{1}{4}$	$\frac{1}{4}$	0
4	0	$\frac{1}{4}$	$\frac{1}{4}$	$\frac{1}{4}$	$\frac{1}{4}$	0
5	0	$\frac{1}{4}$	$\frac{1}{4}$	$\frac{1}{4}$	$\frac{1}{4}$	0
6	$\frac{1}{4}$	$\frac{1}{16}$	$\frac{1}{16}$	$\frac{1}{16}$	$\frac{1}{16}$	$\frac{1}{2}$

(b)

Table 4.1: (a) Adjacency matrix of the discrete motion graph on Figure 4.1. (b) Matrix of transition probabilities of the discrete motion graph on Figure 4.1

the starting positions α_6 and α_5 is marked in bold red stroke. Without loss of generality, assume that the starting positions are not link positions (notice that if some starting positions are on a communication link then taking as starting positions the trajectory points after (or before) ϵ units of time we get an equivalent synchronization schedule). Table 4.1a shows the adjacency matrix of G. Note that the diagonal entries (i, i) have value equal to 1 because in one unit of time a robot can make a tour along its trajectory and come back to the same starting position. We call G the *discrete motion graph* of the system.

Due to synchronization, if a robot is at some position in α then all other robots are also at some position in α (regardless the strategy being used). Moreover, if a robot is at a position in α then, after one unit of time, it will also be at some position in α. The movement of a robot in $\mathcal{F}$ can be modeled by a 'walk' on G and one step of the 'walker' on G corresponds to one time unit on $\mathcal{F}$. Also, given

a sequence of edges (a path) on G traversed by a walker, we are able to get the traversed path by the corresponding robot in $\mathcal{F}$. Thus, the behavior of k available robots in $\mathcal{F}$ is modeled by k simultaneous walkers on G.

The random strategy can be modeled using standard random walks on G. This does not happen for quasi-random- or deterministic- strategies, since the movement of an agent in G depends on the motion of the other agents of the system (Remark 4.1.2). The following section focus on the study of the random strategy using random walks on the graph G.

4.2.2 Randomized SCS's

This section focus on the theoretical study of the **random** strategy using random walks on the graph G. Let $G = (V, E)$ be the discrete motion graph of a R-SCS with synchronization schedule (α, δ). Let α_i and α_j be the starting positions of trajectories C_i and C_j, respectively. If $(i, j) \in E$, there exists a path P of length 2π from α_i to α_j. Let c_{ij} be the number of communication links traversed by P. For an illustration, in Figure 4.1 the path from α_6 to α_5 traverses four communication links and $c_{6,5} = 4$. The transition matrix of the process $M = (p_{ij})_{i,j \in V}$ is defined as follows:

$$M_{ij} = \begin{cases} (1/2)^{c_{ij}} & \text{if } (i,j) \in E \\ 0 & \text{otherwise.} \end{cases}$$

Table 4.1b shows the transition matrix associated to the discrete motion graph of the SCS shown in Figure 4.1. Using the definition of M and taking into account that in every communication link there are two possible paths to follow, we immediately derive that:

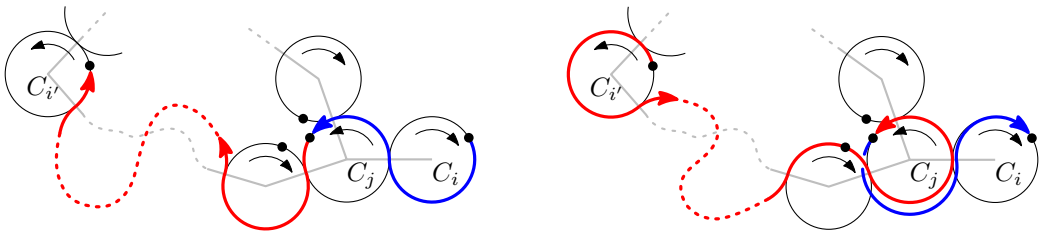
Property 4.2.1. *The matrix M is a right stochastic matrix, i.e., $\sum_j M_{ij} = 1$.*

Lemma 4.2.2. *The transition matrix M associated to a random walk on a discrete motion graph is* doubly stochastic. *That is:*

$$\sum_i M_{ij} = \sum_j M_{ij} = 1.$$

Proof. Let $S = (\alpha, \delta)$ be the synchronization schedule used in the system. By Property 4.2.1 we have $\sum_j M_{ij} = 1$. Now, let us prove that $\sum_i M_{ij} = 1$.

Let $S' = (\alpha, -\delta)$ be the synchronization schedule that results from reversing the orientation of every circle and maintaining the synchronized starting points. Let $G' = (V, E')$ be discrete motion graph of the system using S' and let M' be the transition matrix associated to a random walk on G'. Note that G and G' have the same set of vertices V and $(i, j) \in E$ if and only if $(j, i) \in E'$. Also note that $M_{ij} = M'_{ji}$ and then $M' = M^\intercal$. By Property 4.2.1 we know that $\sum_j M'_{ij} = 1$, that is, each row of M' adds up to 1. Therefore, each column of M adds up to 1 and the result follows. □



(a) (b)

Figure 4.2: Illustration of the proof of Lemma 4.2.3. (a) In red: path from α_j to $\alpha_{i'}$. In blue: path from α_i to α_j. (b) In red: path from $\alpha_{i'}$ to α_j. In blue: path from α_j to α_i.

Lemma 4.2.3. *The discrete motion graph of a synchronized system is strongly connected.*

Proof. We can prove this lemma by induction on the number of trajectories of the system. It is easy to check that the lemma holds for a synchronized system with one or two trajectories. Suppose, as inductive hypothesis, that the lemma holds for any synchronized system of $n \geq 2$ trajectories.

Let $\mathcal{F}$ be a synchronized system with $n+1$ trajectories. Let G be the discrete motion graph of $\mathcal{F}$. Let T be a spanning tree of the communication graph of $\mathcal{F}$. Let C_i and C_j be two trajectories of the system such that C_i corresponds to a leaf of T and C_j is adjacent to C_i in T, see Figure 4.2. Notice that by removing C_i from the system and keeping the starting positions and travel directions in the others trajectories we get a synchronized system $\mathcal{F}'$ with n trajectories. Let G' be the discrete motion graph of $\mathcal{F}'$. Let i' be an arbitrary vertex of G'. By inductive hypothesis there two paths in G', one from i' to j and another from j to i'. Then, there exist paths in $\mathcal{F}'$ from $\alpha_{i'}$ to α_j and viceversa, Figure 4.2 shows these paths using red strokes in subfigures (a) and (b), respectively. Observe that, in $\mathcal{F}$ there are paths of length 2π from α_i to α_j and viceversa (these paths are shown using blue color in Figure 4.2, (a) and (b), respectively). Therefore, the arcs (i, j) and (j, i) are in G. As a consequence, the path from i' to j (resp. from j to i') in G' can be extended using the edge (j, i) (resp. (i, j)) and the lemma is fulfilled. □

From the previous lemma is directly deduced that:

Corollary 4.2.4. *A random walk in a discrete motion graph is* irreducible.

Also, from the previous lemma and taking into account that $M_{ii} > 0$ for all i we get that:

Corollary 4.2.5. *A random walk in a discrete motion graph is* aperiodic.

Now, from Corollary 4.2.5, Lemma 4.2.3 and Lemma 4.2.2 we arrive to the main result of this section:

Theorem 4.2.6. *A random walk on a discrete motion graph has a stationary distribution π^* and π^* is uniform. That is, $\pi^*(i) = 1/n$ for all $1 \leq i \leq n$, where n is the number of trajectories in the system.*

From Theorem 4.2.6 and using the properties of a synchronized system we have:

Corollary 4.2.7. *Let $\mathcal{F}$ be a R-SCS. Let (α, δ) denote the synchronization schedule used on $\mathcal{F}$. Assuming that the starting position of a robot is picked with uniform probability in $\alpha = \{\alpha_1, \ldots, \alpha_n\}$, then after $t \in \mathbb{R}^+$ units of time, the robot is at some point of $\alpha' = \{\alpha_1 + 2\pi\delta_1 t, \ldots, \alpha_n + 2\pi\delta_n t\}$ with uniform probability. Moreover, (α', δ) is a new synchronization schedule (which is equivalent[1] to (α, δ)).*

Let $\mathcal{F}$ be a R-SCS and let $G = (V, E)$ be its discrete motion graph. Knowing that the stationary distribution of a random walk on G is uniform, in the following two subsections we present some theoretical results about the idle-time and the isolation-time of $\mathcal{F}$. Obtaining theoretical results for the broadcast-time for our model seems to be hard and it remains open for future works. Indeed, previous works are limited to specific graph families, namely grids (using a protocol different from ours, since they consider all possible directions at each vertex) and random graphs. See [55] and references therein for a recent work on the broadcast time.

The idle-time

In this subsection we show results on the idle-time of a R-SCS with k robots.

Theorem 4.2.8. *Suppose that k agents are doing a random walk on a discrete motion graph $G = (V, E)$ and their starting vertices are taken uniformly on V. Let v an arbitrary vertex of V. The expected number of robots that visit v during an interval of $\lceil \frac{n}{k} \rceil$ steps is at least one ($n = |V|$).*

Proof. Let $t_1, \ldots, t_c$ be an arbitrary sequence of $c = \lceil \frac{n}{k} \rceil$ steps of the random walk ($t_i = t_1 + i - 1$ for all $1 < i \leq c$). By Theorem 4.2.6, at the step t_i ($1 \leq i \leq c$) each robot is at v with probability $1/n$. For every $1 \leq i \leq c$ and every $1 \leq j \leq k$, let X_{ij} be the random variable given by:

$$X_{ij} = \begin{cases} 1 & \text{if the } j\text{-th drone is at } v \text{ at step } t_i, \\ 0 & \text{otherwise.} \end{cases}$$

Therefore,

$$E[X_{ij}] = \frac{1}{n}.$$

Let

$$Y = \sum_{i=1}^{c} \sum_{j=1}^{k} X_{ij}.$$

[1] See Definition 2.2.10 of Chapter 2.

Note that the expected number of robots that visit v during the given interval of steps is equal to $E[Y]$. By linearity of expectation

$$E[Y] = \sum_{i=1}^{c}\sum_{j=1}^{k} E[X_{ij}] = \sum_{i=1}^{c}\sum_{j=1}^{k} \frac{1}{n} = c\frac{k}{n} = \left\lceil \frac{n}{k} \right\rceil \frac{k}{n} \geq 1.$$

□

From the previous theorem we obtain:

Corollary 4.2.9. *Suppose that k agents are doing a random walk on a discrete motion graph $G = (V, E)$ and their starting vertices are taken uniformly on V. For all $v \in V$, let I_v be the random variable that indicates the number of steps between two consecutive visits to v. The expected value of I_v is $E[I_v] \leq \lceil \frac{n}{k} \rceil$, where $n = |V|$.*

Taking into account that one step in the discrete motion graph corresponds to one time unit and, by using Corollary 4.2.7 and Theorem 4.2.8, we deduce the following conclusion:

Corollary 4.2.10. *Let $\mathcal{F}$ be a R-SCS where k robots are operating. If the robots start at randomly chosen positions then the idle-time is at most $\lceil \frac{n}{k} \rceil$, where $n = |V|$.*

The result above assumes that the robots are uniformly located at the beginning and that the system lies in the stationary distribution. However, in practical applications the robots are all deployed at a given position, for instance in the trajectories nearest to the boundary of the global region. In this case, we need to study the time it takes for the stationary distribution to be reached by the system, that is, the *mixing time*. Informally, the mixing time is the time needed for a random walk to reach its stationary distribution, or, more specifically, the number of steps before the distribution of a random walk is *close* to its stationary distribution. In the experiments, we will show that the mixing time in our model is really small for grids in real scenarios and we can say that, after a few units of time (5 for a 5×5 grid, i.e., for 25 trayectories), the idle-time of the random strategy for k drones is at most $\lceil \frac{n}{k} \rceil$.

The isolation-time

A *meeting* occurs when two robots arrive at the same time at a common communication link between their trajectories. We want to study the expected time a robot is isolated, that is, the expected time between two consecutive meetings of a robot. We study this measure theoretically in the discrete motion graph using resources from random walks as in the previous section. Therefore, in this theoretical study we consider a meeting between two agents when they visit a same vertex of the discrete motion graph at the same time.

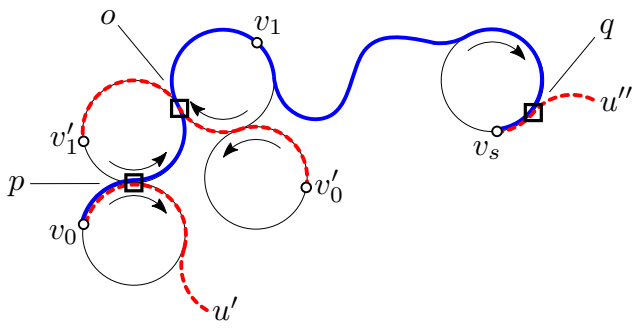


Figure 4.3: The points $v_0, v_1, \ldots, v_s, v_0', v_1'$ are the corresponding starting positions on their circles. The path $v_0, v_1, \ldots, v_s$, traversed by a robot u, is shown using blue solid stroke. The robot u' departs with u from v_0 at time t and they separate their paths at the communication link p. The robot u'' meets u at the communication link q and they arrive at v_s traveling together. If a robot u''' departs from v_0' at time t following the red dashed stroke that ends at v_1', then, u and u''' meet in the communication link o without visiting any starting position together.

In the following, we describe the difference between the isolation-time measure on a R-SCS and its corresponding discrete communication graph. Let $\mathcal{F}$ be a R-SCS and let G be its corresponding discrete motion graph. Suppose that, in G, an agent u has two consecutive meetings at times t and $t+s$. Thus, agent u has been isolated by s time units in G. Let $v_0, \ldots, v_s$ be the consecutive sequence of vertices of G visited by u in this time interval. Let u' and u'' be the agents that meet u in v_0 and v_s, respectively. Having into account that the vertices of G are not communication links, then, in $\mathcal{F}$, u and u' separate their paths in a communication link p traversed by them at some time t' such that $t < t' < t+1$, see Figure 4.3. Similarly, u and u'' meet in a communication link q of $\mathcal{F}$ at some time t'' such that $t+s-1 < t'' < t+s$, see Figure 4.3. Therefore, in $\mathcal{F}$, u has been isolated by less than s time units. Also, notice that, if two robots in $\mathcal{F}$ meet in a communication link (o in Figure 4.3) coming from two different starting positions (v_0 and v_0' in Figure 4.3) and, after the meeting, they go toward different starting positions (v_1 and v_1' in Figure 4.3), then, this meeting is not reported in G.

From these observations, the following remark is straightforward deduced:

Remark 4.2.11. *The isolation-time of a robot in a R-SCS is always less than the isolation-time in the corresponding discrete motion graph.*

Theorem 4.2.12. *Suppose that k agents are doing a random walk on a discrete motion graph $G = (V, E)$ and their starting vertices are taken uniformly on V. Then, the expected number of times that an agent meets another one at some vertex of V during an interval of $\left\lceil \frac{n^{k-1}}{n^{k-1} - (n-1)^{k-1}} \right\rceil$ steps is at least one.*

Proof. Let $t_1, \ldots, t_c$ be an arbitrary sequence of $c = \left\lceil \frac{n^{k-1}}{n^{k-1} - (n-1)^{k-1}} \right\rceil$ steps of a random walk ($t_i = t_1 + i - 1$ for all $1 < i \leq c$) performed by an agent u. By Theorem 4.2.6, we deduce that u is alone at a vertex v of V at step t_i with probability: $\frac{1}{n}\left(\frac{n-1}{n}\right)^{k-1} = \frac{(n-1)^{k-1}}{n^k}$. And, u is not at v at step t_i with probability $\frac{n-1}{n}$. Thus, the probability of u being with at least another robot at v at step t_i is given by:

$$1 - \frac{(n-1)^{k-1}}{n^k} - \frac{n-1}{n} = \frac{n^{k-1} - (n-1)^{k-1}}{n^k}.$$

Let $X_{i,v}$ be the random variable given by:

$$X_{i,v} = \begin{cases} 1 & \text{if } u \text{ meets some robot at } v \text{ at step } t_i, \\ 0 & \text{otherwise.} \end{cases}$$

It easy to see that

$$E[X_{i,v}] = \frac{n^{k-1} - (n-1)^{k-1}}{n^k}.$$

Let

$$Y = \sum_{i=1}^{c} \sum_{v \in V} X_{i,v}.$$

Now, the expected number of times that u meets another agent at some vertex of V during the given sequence of steps is equal to $E[Y]$ and

$$E[Y] = \sum_{i=1}^{c} \sum_{v \in V} E[X_{i,v}] = \sum_{i=1}^{c} \sum_{v \in V} \left(\frac{n^{k-1} - (n-1)^{k-1}}{n^k} \right) \geq 1.$$

□

From this theorem we arrive to:

Corollary 4.2.13. *Suppose that k agents are doing a random walk on a discrete motion graph $G = (V, E)$ and their starting vertices are taken uniformly on V. For every agent, the expected number of steps between two consecutive meetings is at most $\left\lceil \frac{n^{k-1}}{n^{k-1} - (n-1)^{k-1}} \right\rceil$.*

Now, extending the previous result to a R-SCS (using Remark 4.2.11), we have:

Corollary 4.2.14. *Let $\mathcal{F}$ be a R-SCS where k robots are operating. If the robots start at randomly chosen positions then, for every robot, the expected time between two consecutive meetings is less than $\left\lceil \frac{n^{k-1}}{n^{k-1} - (n-1)^{k-1}} \right\rceil$.*

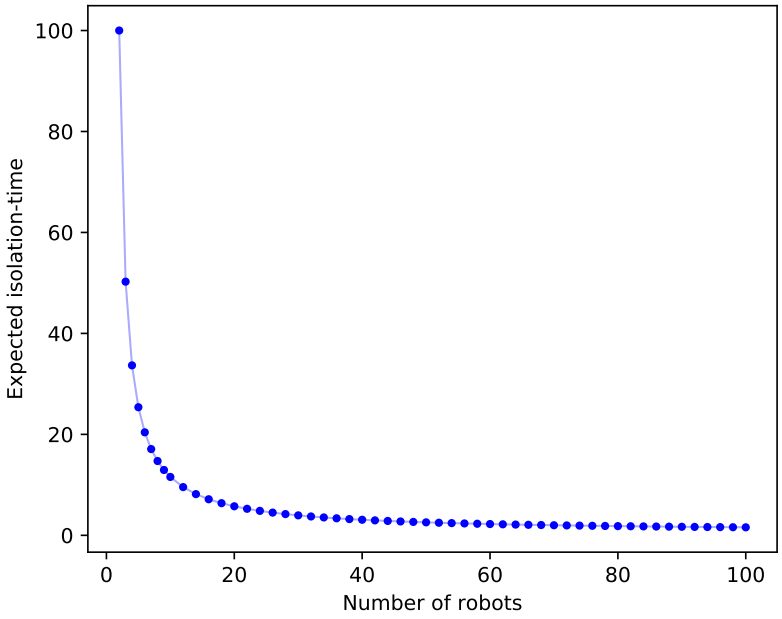


Figure 4.4: Behavior of the function $I(k) = \left\lceil \frac{n^{k-1}}{n^{k-1}-(n-1)^{k-1}} \right\rceil$, the expected isolation-time for $2 \leq k \leq 100$ robots considering a scenario of $n = 100$ trajectories.

Observe that Corollary 4.2.14 says that a robot meets another robot in an interval of time of length at most $\left\lceil \frac{n^{k-1}}{n^{k-1}-(n-1)^{k-1}} \right\rceil$. This implies that for only two robots in the system, they meet at most every n times unit (consider $k = 2$ in the formula).

Figure 4.4 shows the behavior of the expected time when k is increasing. The behavior of this function says that a small number $k << n$ of drones is enough to get a good result with the random strategy .

4.3 Experimental results

In this section, we evaluate the validity of the proposed random strategies comparing the values of idle-time, isolation-time and broadcast-time with the results obtained using the deterministic strategy. Also, we show that the bounds we proved for idle-time and isolation-time are very tight and evaluate the mixing time to explore the rate of convergence of the system to the uniform distribution. These experiments were implemented in Python3.6 using NumPy 1.15.2. The code

is available in GitHub[2].

4.3.1 Experiments description

Typically, in surveillance tasks with small drones, the map is split into a grid of small cells. Thus, we performed several experiments on grid graphs of sizes 10×10, 15×15, 20×20 and 30×30. For every one of these grids of $N\times N$ size, we simulated the random strategies with k robots, $1 \leq k < N^2$. And, for every grid and every value k, we have performed 10 repetitions of the experiment, choosing the starting position of each robot uniformly at random from the synchronized positions. Each experiment in an $N \times N$ grid ran for $4N^2$ units of time. In the following, $E^{(i)}_{N,k}$ denotes the i-th repetition of an experiment in a grid of size $N \times N$ using k robots.

Let $\mathcal{F}$ be a synchronized $N \times M$ grid shaped system. Let $\mathcal{W}_{N,M}$ be an auxiliary graph, that we call *walking graph*, whose vertices are the communication links (the edges between neighboring circles of $\mathcal{F}$) and the $2(N+M)$ touching points between the trajectories and an imaginary box circumscribed on $\mathcal{F}$, see Figure 4.5. There is a directed edge (v, w) in $\mathcal{W}_{N,M}$ if there exists an arc of length $\pi/2$ between the points corresponding to v and w following the assigned travel directions. In the following, let $V(\mathcal{W}_{N,M})$ and $E(\mathcal{W}_{N,M})$ denote the set of vertices and arcs of $\mathcal{W}_{N,M}$, respectively. Each robot moves on $\mathcal{F}$ following the travel directions and can perform a shifting operation only at the communication links. Thus, any (not necessarily simple) path in $\mathcal{W}_{N,M}$ corresponds to a valid sequence of movements of a robot in $\mathcal{F}$ where a step in $\mathcal{W}_{N,M}$ corresponds to $1/4$ of a time unit in $\mathcal{F}$. In this way, the movement of an agent in $\mathcal{W}_{N,M}$ (following the arcs in $E(\mathcal{W}_{N,M})$) emulates a valid sequence of movements of a robot in $\mathcal{F}$. We performed the experiments on the walking graph $\mathcal{W}_{N,N}$. From now on, for simplicity we denote the walking graph by $\mathcal{W}_N$.

4.3.2 Experiments for the idle-time

Note that every point at a circle of an $N \times N$ grid SCS is at some arc of $\mathcal{W}_N$. Hence, the idle-time of an $N \times N$ grid SCS using k robots is the idle-time of the arcs of $\mathcal{W}_N$ using k robots. In order to measure this value, we do the following: in each experiment $E^{(i)}_{N,k}$, for every arc $e \in E(\mathcal{W}_N)$, we count the number of times that e is traversed by a robot and when it is traversed. Having all the visits to an arc e, we can compute the average time $\texttt{idle}^{(i)}_e(k)$ between consecutive passes through the arc e in the experiment $E^{(i)}_{N,k}$.

Let $\texttt{idle}_e(k) = \frac{1}{10}\sum_{i=1}^{10} \texttt{idle}^{(i)}_e(k)$ denote the average time between consecutive passes through the arc e all over the 10 repetitions of the experiment using k robots. Finally, in order to compare the idle-time of these strategies having into account

[2]https://github.com/varocaraballo/random-walks-synch-sq-grid

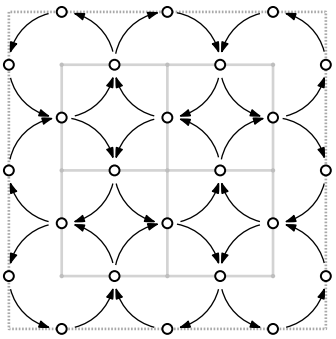


Figure 4.5: The walking graph of the 3×3 grid SCS shown in Figure 3.1(b). The imaginary box where the system is inscribed is drawn using dotted stroke. The vertices of the walking graph are the non-solid points. Notice that the directed arcs between these nodes follow the travel directions assigned in Figure 3.1(b).

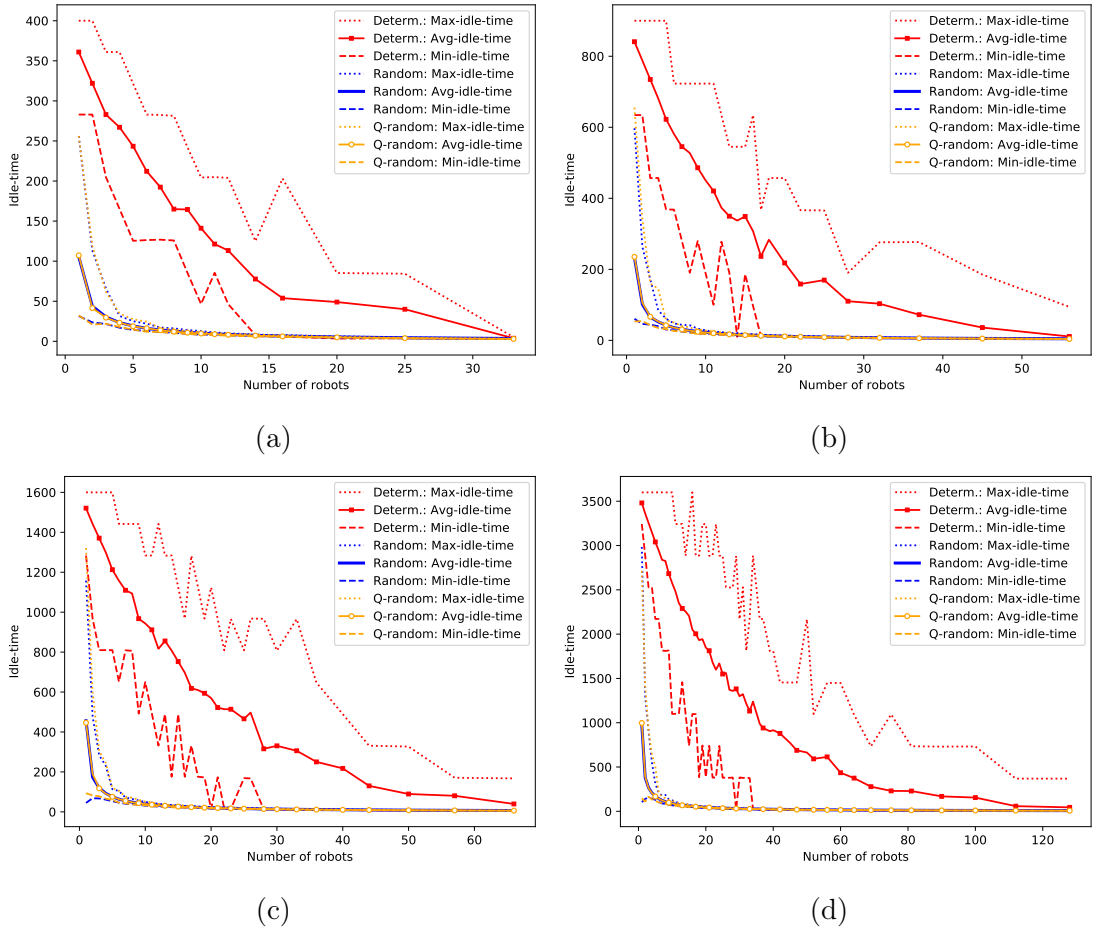


Figure 4.6: Comparison of the idle-time obtained in the experiments using the strategies random, quasi-random (Q-random) and deterministic: (a) 10×10 grid SCS, (b) 15×15 grid SCS, (c) 20×20 grid SCS and (d) 30×30 grid SCS.

the worst case, average case and best case we define the functions

$$\texttt{max\_idle}_{\mathcal{W}_N}(k) = \max_{e \in E(\mathcal{W}_N)} \{\texttt{idle}_e(k)\},$$
$$\texttt{avg\_idle}_{\mathcal{W}_N}(k) = \frac{1}{|E(\mathcal{W}_N)|} \sum_{e \in E(\mathcal{W}_N)} \texttt{idle}_e(k), \text{and}$$
$$\texttt{min\_idle}_{\mathcal{W}_N}(k) = \min_{e \in E(\mathcal{W}_N)} \{\texttt{idle}_e(k)\}, \text{ respectively.}$$

We have computed these three functions using the three mentioned strategies and the results are shown in Figure 4.6. The three functions `max_idle`, `avg_idle` and `min_idle` are shown using dotted, solid and dashed stroke respectively. These functions are shown in three different colors: blue, orange and red for random, quasi-random and deterministic strategy, respectively.

Figures 4.6(a), 4.6(b), 4.6(c) and 4.6(d) correspond to grid SCSs of sizes 10×10, 15×15, 20×20 and 30×30, respectively. In every subfigure we show the evolution of `max_idle`, `avg_idle` and `min_idle` for the three strategies. Note that, in the charts of Figure 4.6 we do not show the behavior of the functions until the maximum tested value of k (which is N^2) because we want to emphasize the differences between the three tested strategies and these differences are evident for low values of k (after a certain value of k, the functions are similar for all the strategies). Note that, when using low values of k, there is a huge difference between the deterministic strategy and the other ones. This difference is not surprising because from Lemma 3.3.8, we know that an $N \times N$ grid SCS graph has N *rings* and to cover everything using the deterministic strategy, at least one robot per ring is required, Lemma 3.3.2. Therefore, with less than N robots is not possible to cover every point and we get a very high idle-time. In these cases the idle-time is bounded by the duration of the simulation ($4N^2$), although, what really happens is that some points are never visited. Note that, because the starting positions are taken randomly, even if $k \geq N$, some rings could be empty of robots. Therefore, all the points involved in these rings are uncovered.

Also, note that the `max_idle` is not $4N^2$ for every $k < N$. That is because the value $\texttt{idle}_e(k)$ is amortized among the 10 repetitions of the experiment (in some experiment $E^{(i)}_{N,k}$ a ring r may be empty of robots and then $\texttt{idle}^{(i)}_e(k) = 4N^2$ for every arc e involved in r, but in some other experiment $E^{(j)}_{N,k}$, the ring r may have $c > 0$ robots, thus, $\texttt{idle}^{(j)}_e(k) = N/c$ for every arc e involved in r).

Notice that with the random or quasi-random strategies we get a very similar behavior. In fact, a very good result because the function $\texttt{avg\_idle}_{\mathcal{W}_N}$ is close to the function $f(k) = N^2/k$ and this is the best possible behavior taking into account that we have $2\pi N^2$ length of trajectory (the sum of the N^2 circles) to cover and k robots moving at 2π units of length per time unit. Moreover, by using one of these random strategies, the function $\texttt{avg\_idle}_{\mathcal{W}_N}$ starts decreasing rapidly and, around to the value $k = N = \sqrt{n}$ (recall $n = N^2$ in these grid SCSs), the

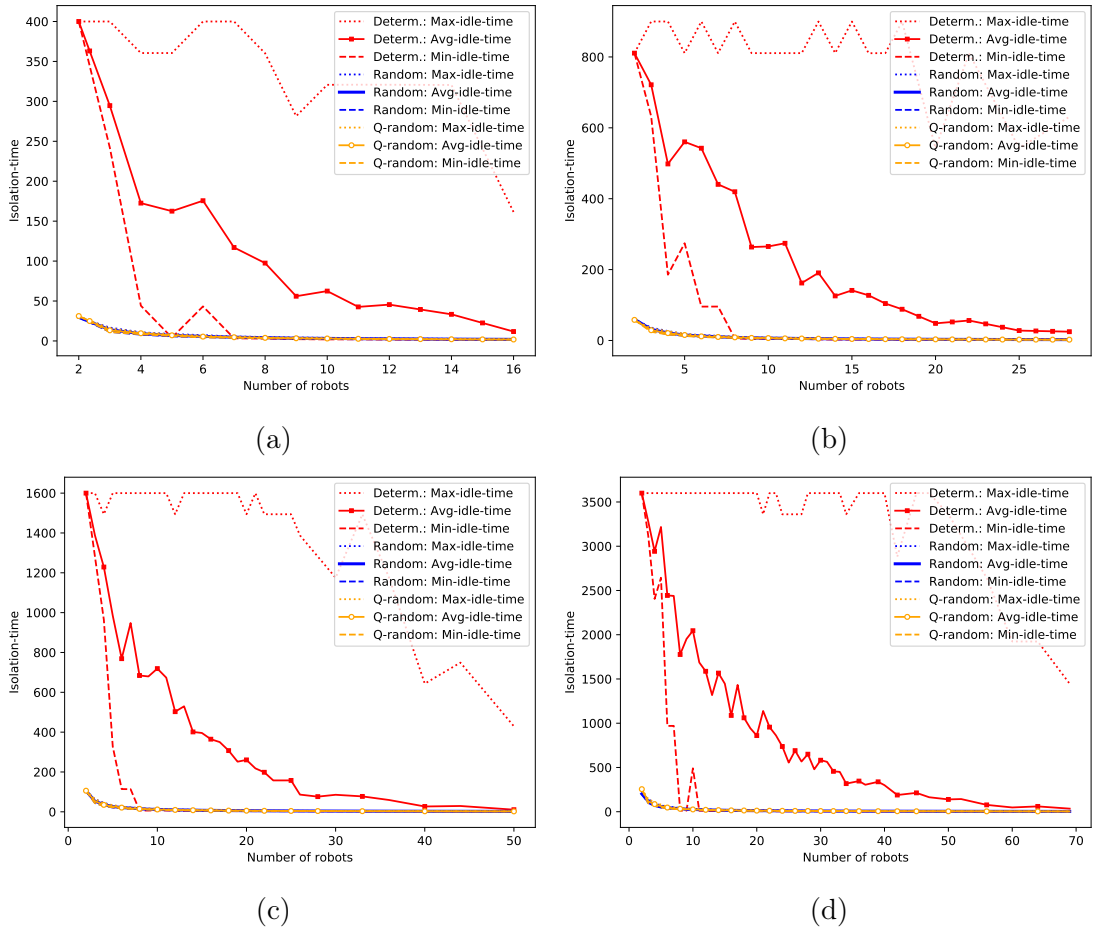


Figure 4.7: Comparison of the isolation-time obtained in the experiments using the strategies random, quasi-random (Q-random) and deterministic: (a) 10×10 grid SCS, (b) 15×15 grid SCS, (c) 20×20 grid SCS and (d) 30×30 grid SCS.

decreasing rate is notably reduced. This behavior indicates that, when one of the random strategies is being used in a SCS with $k \geq N$ robots, the addition of more robots to the system does not represent a significant improvement of the idle-time. Another conclusion that we can extract from the experiments is that, using any of the two random strategies with very few robots ($k \approx N$), the idle-time is really good. However, to get good results using the deterministic strategy we need, with *high probability*, a much larger number of robots.

4.3.3 Experiments for the isolation-time

Note that every meeting between two robots in an $N \times N$ grid SCS takes place at one of the vertex of the walking graph $\mathcal{W}_N$. Hence, in an $N \times N$ grid SCS using a set U of k robots, the isolation-time of the system is the expected time

between two consecutive meetings at a vertex of $\mathcal{W}_N$ using k robots. In order to measure this value, we do the following: in each experiment $E^{(i)}_{N,k}$, for every robot $u \in U$, we count the number of times that u meets some other robot and when these meetings occur. Having all the meetings of u we can compute the average time $\texttt{isolation}^{(i)}_u(k)$ between consecutive meetings of u in the experiment $E^{(i)}_{N,k}$.

In order to compare the isolation-time of the strategies having into account the worst case, average case and best case, we define the functions:

$$\texttt{max\_isolation}^{(i)}_{\mathcal{W}_N}(k) = \max_{u \in U}\{\texttt{isolation}^{(i)}_u(k)\},$$
$$\texttt{avg\_isolation}^{(i)}_{\mathcal{W}_N}(k) = \frac{1}{k}\sum_{u \in U}\texttt{isolation}^{(i)}_u(k),$$
$$\texttt{min\_isolation}^{(i)}_{\mathcal{W}_N}(k) = \min_{u \in U}\{\texttt{isolation}^{(i)}_u(k)\},$$

respectively. Then, we compute the averaged values all over the 10 repetitions of the experiments:

$$\texttt{max\_isolation}_{\mathcal{W}_N}(k) = \sum_{i=1}^{10}\frac{\texttt{max\_isolation}^{(i)}_{\mathcal{W}_N}(k)}{10},$$
$$\texttt{avg\_isolation}_{\mathcal{W}_N}(k) = \sum_{i=1}^{10}\frac{\texttt{avg\_isolation}^{(i)}_{\mathcal{W}_N}(k)}{10},$$
$$\texttt{min\_isolation}_{\mathcal{W}_N}(k) = \sum_{i=1}^{10}\frac{\texttt{min\_isolation}^{(i)}_{\mathcal{W}_N}(k)}{10}.$$

We have computed the values of these three global functions using the three tested strategies and the results are shown in Figure 4.7. The three functions `max_isolation`, `avg_isolation` and `min_isolation` are shown using dotted, solid and dashed stroke, respectively. These functions are shown in three different colors: blue, orange and red for random, quasi-random and deterministic strategy, respectively.

Figures 4.7(a), 4.7(b), 4.7(c) and 4.7(d) correspond to grid SCSs of sizes 10×10, 15×15, 20×20 and 30×30, respectively. In every subfigure, we show the evolution of the `max_isolation`, `avg_isolation` and `min_isolation` for the three strategies for $k \geq 2$ (note that a single robot in the system is always isolated). As in the previous subsection, Figure 4.7 does not show the behavior of the functions until the maximum tested value of k because the differences between the strategies are much more evident for low values of k (after a certain value of k, the functions are similar between them for all the strategies). Note that, using low values of k, the deterministic strategy is much worse than the random ones. Notice that for two robots the values of `max_isolation`, `avg_isolation` and `min_isolation` are the same (that is, the three functions start at the same point for every strategy).

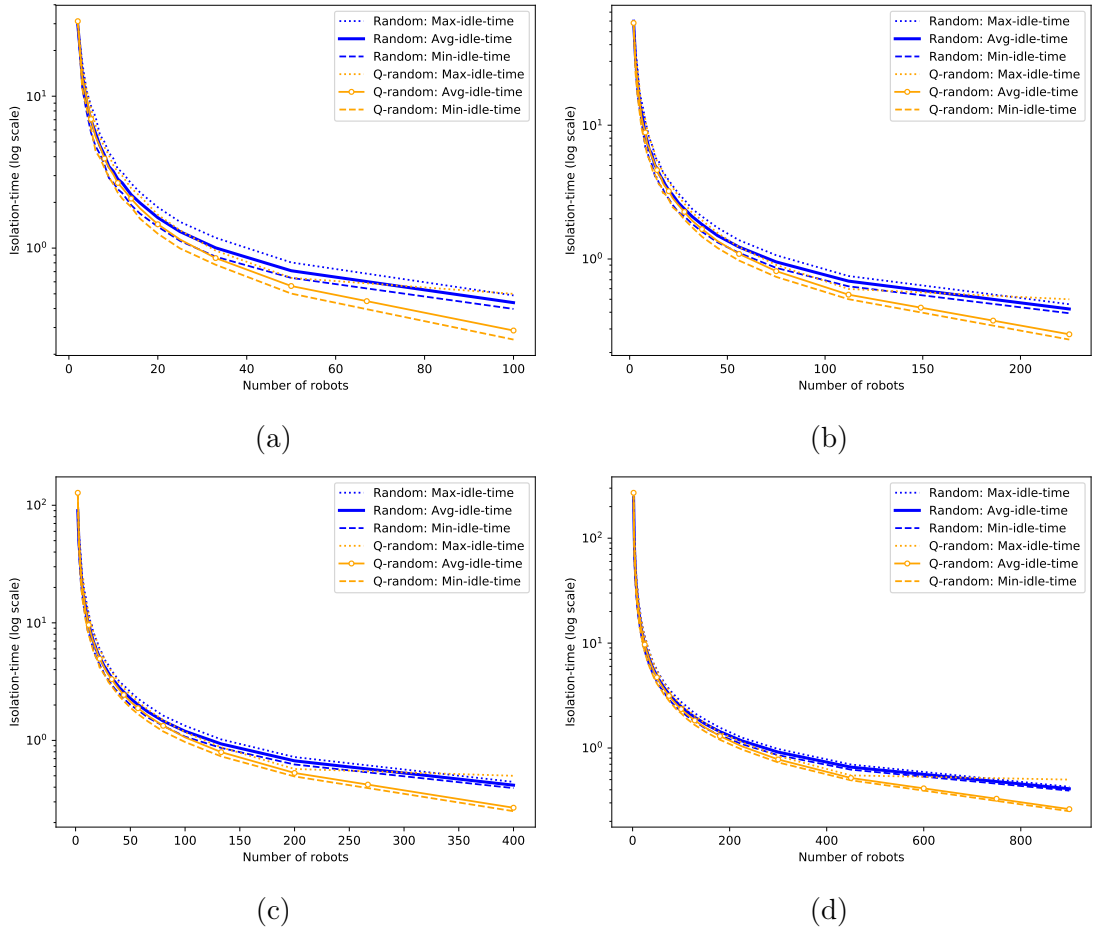


Figure 4.8: Comparison of the isolation-time obtained in the experiments using the strategies random and quasi-random (Q-random): (a) 10×10 grid SCS, (b) 15×15 grid SCS, (c) 20×20 grid SCS and (d) 30×30 grid SCS.

This behavior is because when a robot has a meeting, it is with the other one, so, they both have exactly the same statistics. Therefore, the average, maximum and minimum isolation times of them have the same value.

From Theorem 3.4.45, we know that in an $N \times N$ grid SCS with robots using the deterministic strategy, two robots meet each other if and only if they start in circles of the same row or column. Thus, using $k \ll N^2$ robots, it is very probable that a robot starts in a circle whose row and column are different from the rows and columns of any other robot in the system. Because of this, the values of `max_isolation` are so high for small values of k. Also, note that using a relative small number of robots, but greater than 2, the probability of at least two of them start at the same column or the same row is high. Because of this, the `min_isolation` function decreases quickly. Therefore, using the deterministic strategy there is a huge difference between the functions `max_isolation`, `avg_isolation` and `min_isolation`. Finally, notice that `avg_isolation` is closer to `min_isolation` than to `max_isolation`. This indicates that using a small group of robots there is a high probability that one of them is isolated but, it is very probable that the number of non-isolated robots is greater than the number of isolated robots.

In Figure 4.7, it is difficult to observe the behavior of the random strategies due to the high values of isolation-time using the deterministic strategy. Notice that we get a very similar behavior using any of the randomized strategies, which it is much better than using the deterministic strategy. We have added Figure 4.8 to compare the values of isolation-time using the random strategies. We have used a logarithmic scale in the Isolation-time axis. Notice that, the greater the number of robots is, the better Quasi-random strategy is with respect to Random strategy. Another conclusion that we can extract from Figure 4.8 is that, using any of the random strategies, with very few robots respect to the number of trajectories (cells), i.e. $k \approx N = \sqrt{n}$, we obtain a very good value of isolation-time. However, to get good results using the deterministic strategy we need, with *high probability*, a much larger number of robots (see Figure 4.7). Moreover, the value $k = N$ marks a threshold in the function `avg_isolation` when one of the two random strategies is being used because, the addition of more robots to a SCS with $k \geq N$ robots does not represent a significant reduction of the isolation-time.

4.3.4 Experiments for the broadcast-time

In this subsection we study the broadcast-time measure using the three different proposed strategies for grid SCSs. To evaluate this measure, we have elaborated a different experiment. Suppose that we have an $N \times N$ grid SCS $\mathcal{F}$ with $k \geq 2$ ($k \in \mathbb{N}$) robots. In order to estimate the broadcast-time of the system we do the following: In the walking graph $\mathcal{W}_N$ of $\mathcal{F}$ we randomly set the k starting positions of each robot with uniform probability, then we randomly choose a robot to emit a

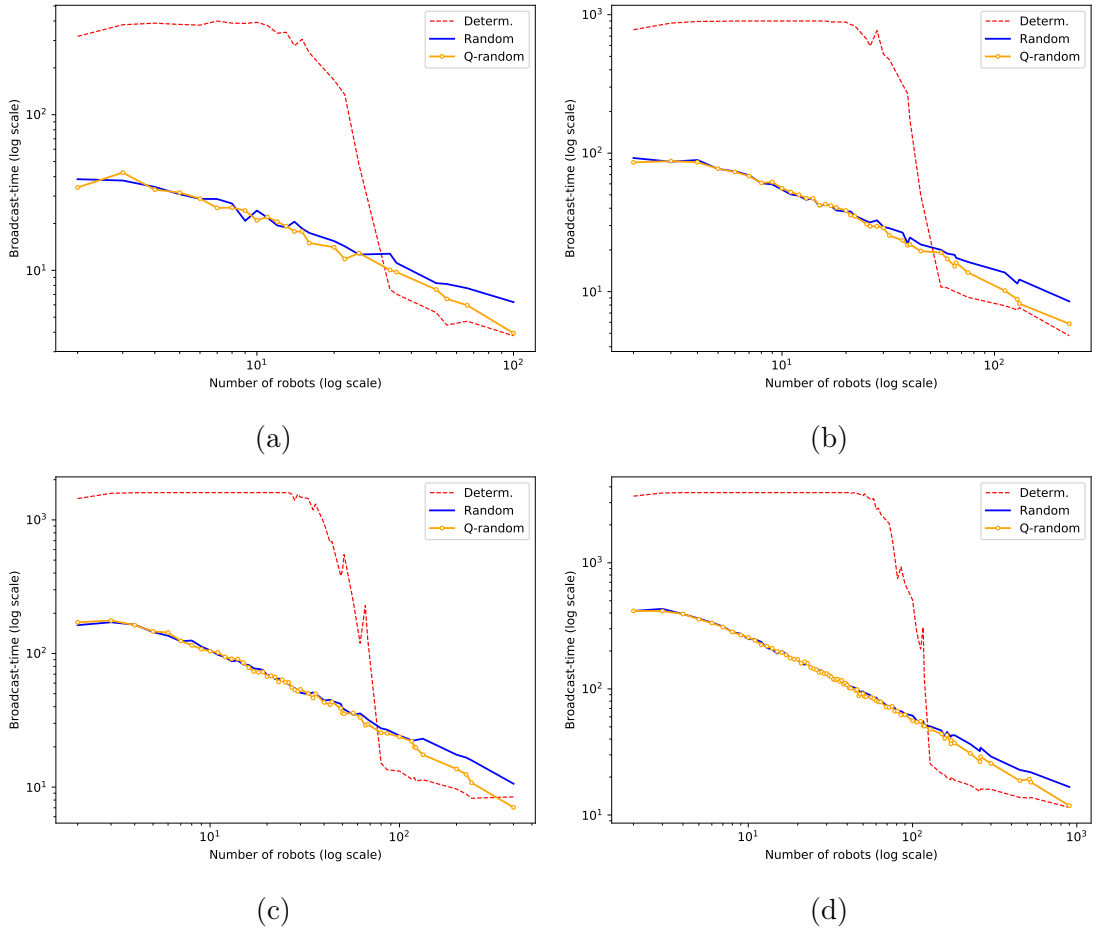


Figure 4.9: Comparison of the broadcast-time obtained in the experiments using the strategies random, quasi-random (Q-random) and deterministic: (a) 10×10 grid SCS, (b) 15×15 grid SCS, (c) 20×20 grid SCS and (d) 30×30 grid SCS.

message. After that, the simulation starts using one of the proposed strategies until all the robots of the team have received the message or $4N^2$ time units have elapsed. Let $E^{(i)}_{N,k}$ and $\texttt{broadcast}^{(i)}_{\mathcal{W}_N}(k)$ denote the i-th repetition of this experiment and its simulation time, respectively. Note that $\texttt{broadcast}^{(i)}_{\mathcal{W}_N}(k) = \min\{4N^2, t\}$ where t is the required time for the message to be known for all the robots in the team. We repeat this experiment N^2 times and we estimate the broadcast-time of $\mathcal{F}$ using k robots by the value:

$$\texttt{broadcast}_{\mathcal{W}_N}(k) = \frac{1}{N^2}\sum_{i=1}^{N^2} \texttt{broadcast}^{(i)}_{\mathcal{W}_N}(k).$$

We have computed the values of the function $\texttt{broadcast}_{\mathcal{W}_N}$ for $2 \leq k < N^2$ and $N = 10$, 15, 20 and 30. The results of our experiments are shown in Figure 4.9 using a logarithmic scale for the broadcast-time and the number of used robots. According to this picture, for small values of k, the random strategies are much better than the deterministic one. As mentioned above, two robots using the deterministic strategy meet each other only if their initial positions are in the same row or the same column (Theorem 3.4.45). Consider now a new auxiliary graph H as follows: add a vertex for every robot and an edge between two robots exists if they are on circles within the same row or the same column. Obviously, it is possible to make a broadcast if and only if H is connected. When the number of robots is small, it is very probable than H is not connected. For this reason, broadcast times are very high by using the deterministic strategy for a small number of robots.

When the number of robots is sufficiently high ($k > N^{4/3}$), the three strategies are more similar. However, we can see that the deterministic strategy is better than the others although the quasi-random strategy gives similar results, see Figure 4.9. The reason of this behavior is that when k is big, the auxiliary graph H is connected and, a broadcast can be completed in approximately N time units. Also, note that, using the quasi-random strategy, the more robots we have the more similar this strategy is to the deterministic one.

Finally, notice that using any of the proposed strategies, the $\texttt{broadcast}_{\mathcal{W}_N}$ function "rises" slightly at the beginning and after that, it decreases. From this, we deduce that at the very beginning, adding robots to the system means an increasing of the broadcast time. This fact means that it is more difficult to complete a broadcast for a bigger team. However, after a certain threshold, the more robots we have, the faster a broadcast is completed. This behavior can be explained because after a certain value of k, an increase on the number of robots helps to spread the message. More specifically, at the beginning only one robot knows the message. If the number of robots is sufficiently large, after a few time units, it will meet someone (as it is shown in the previous subsection), and after that, there are two robots that know the message. Then, after another few time

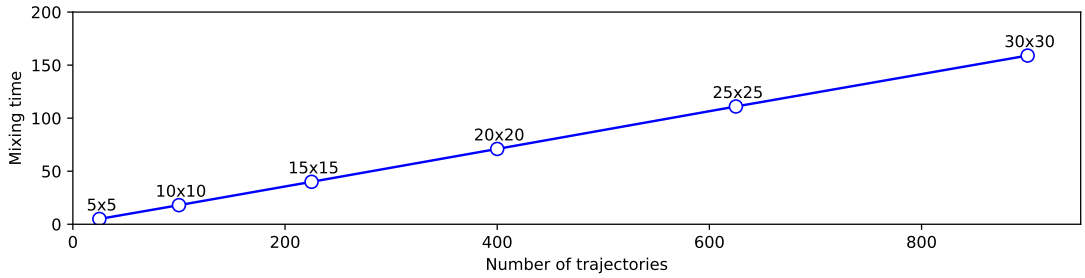


Figure 4.10: Estimated mixing times in R-SCSs of sizes 5×5, 10×10, 15×15, 20×20, 25×25 and 30×30.

units, they will meet more robots and ideally 4 robots would know the messages and so on. Therefore, the more robots carry the message, the faster it is spread.

4.3.5 Mixing time

In this subsection, we experimentally study the mixing time of the distribution corresponding to a random drone in a R-SCS. Our study is based on the discretization of a R-SCS introduced in Section 4.2.1 and the transition matrix presented in Section 4.2.2. The experiment of this section consists on computing the transition matrix M corresponding to a R-SCS $\mathcal{F}$ and then $M^2, M^3, \ldots$ until a matrix close to the uniform distribution is obtained. More precisely, we ask for a matrix M^t such that, for any initial distribution π:

$$||\pi M^t - \pi^*|| < \varepsilon \tag{4.1}$$

where $||\cdot||$ denotes the Euclidean norm of a vector, π^* is the uniform distribution (which is the stationary distribution of the random walk described by a random drone) and ε is a small fixed value. Then, this value t is an estimation of the mixing time.

Notice that $\lim_{t\to\infty} \pi M^t = \pi^*$ for all initial distribution π, then $\lim_{t\to\infty} M^t = M^*$ where M^* is the matrix where every entry is $1/n$. Therefore, $\pi M^* = \pi^*$ and then:

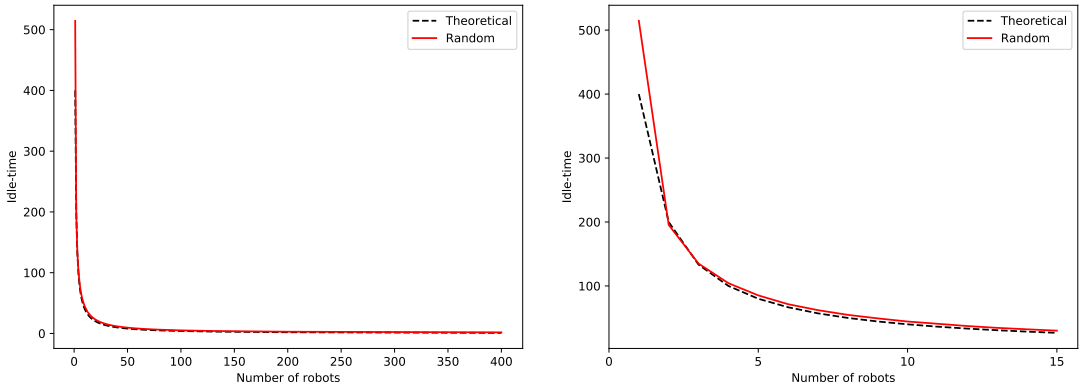
$$||\pi M^t - \pi^*|| = ||\pi M^t - \pi M^*|| = ||\pi(M^t - M^*)|| \leq ||\pi||||M^t - M^*||,$$

where $||M^t - M^*||$ denotes the 2-norm of matrix $M^t - M^*$.

Taking into account that $||\pi|| = 1$ it is deduced that $||\pi M^t - \pi^*|| \leq ||M^t - M^*||$. Then, we look for the minimum value t such that $||M^t - M^*|| < \varepsilon$.

For values of N in $\{5, 10, 15, 20, 25, 30\}$ we build the transition matrix M of an $N \times N$ grid R-SCS and we seek for the smallest t such that $||M^t - M^*|| < \varepsilon = 1/4$. This is a typical value to estimate the mixing time [12, 83, 123].

Figure 4.10 illustrates the obtained results. Notice that the behavior of the mixing time seems to be linear with respect to the number of trajectories. The



(a) (b)

Figure 4.11: Experimental results on idle-time and the theoretical ones using a 20×20 R-SCS. (a) It shows the behavior of the idle-time with respect to $k = 1 \ldots 400$ (number of robots). (b) It shows the same data of (a) but constrained to $k \leq 15$ in order to make visible the differences between the two curves.

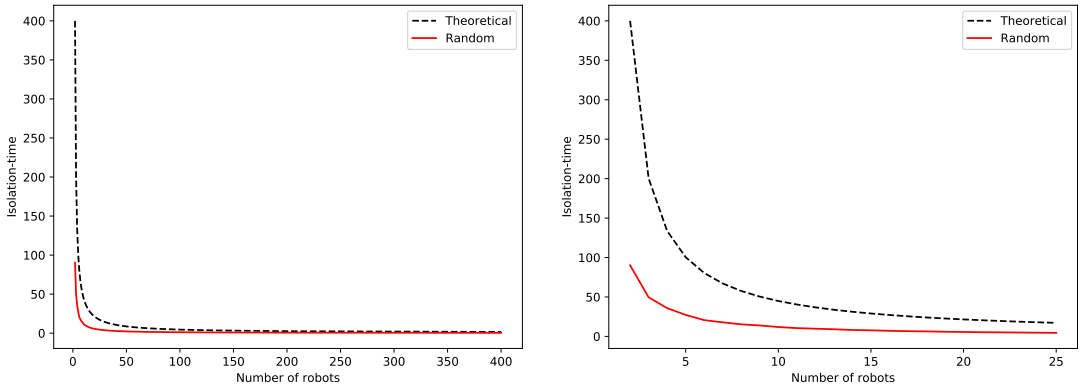
increasing ratio (slope) is 0.17 approximately. That is, for every trajectory added to the system, the mixing time increases by 0.17 units. The obtained estimations of mixing times for R-SCSs of sizes 10×10, 15×15, 20×20 and 30×30 (these are the sizes of the studied R-SCS in previous sections) are 18, 40, 71 and 159, respectively.

4.3.6 Comparison with the theory

In this section we compare the obtained results in the previous experiments using the random strategy and the theoretical results on idle-time and isolation-time of sections 4.2.2 and 4.2.2 respectively. Moreover, a comparison between our experiments on broadcast-time using the random strategy and the theoretical results on regular graphs in [34] is added.

First, we show that the bound obtained for the idle-time is very tight with respect the experiments. See Figure 4.11 where the analytical and the experimental results are shown for a 20×20 grid R-SCS. In Figure 4.11(a) the results on $k = 1...400$ are shown. See that the analytical and the experimental results almost coincide. Figure 4.11(b) shows the same data but constrained to $k \leq 15$ in order to make visible the differences between the two curves which are more evident with few robots. However, notice that, even using a few robots the random strategy have a very good performance, it is almost optimum (recall that n/k is the best possible idle-time in a system formed by k robots in a region divided into n trajectories).

In the comparison of the isolation-time, we found that the theoretical expected value is an upper bound of the average experimental isolation-time as we have



(a) (b)

Figure 4.12: Experimental results on isolation-time and the theoretical ones using a 20×20 R-SCS. (a) It shows the behavior of the isolation-time with respect to $k = 2 \ldots 400$ (number of robots). (b) It shows the same data of (a) but constrained to $k \leq 25$ in order to increase visually the differences between the two curves.

noted in subsection 4.2.2, more specifically, Remark 4.2.11. Figure 4.12 shows the behavior of the theoretical expected value and the average experimental isolation-time in 20×20 grid R-SCS. In Figure 4.12(a) the global behavior of the two curves is shown for $k = 2...400$. Figure 4.12(b) shows the same data but constrained to $k \leq 25$ in order to make easier to see the differences between these curves. Notice that difference between the theoretical and experimental isolation-time decreases when the number of robots increases. This behavior makes sense because the effects of the phenomenon described in the second paragraph of subsection 4.2.2 are more evident when the number of trajectories is large with respect to the number of robots, because there is a lot of room and it is very probable that the robots are spared on the region. However, if we increment the number of robots this effect is *diluted* because the greater the number of robots is, the probability of a robot not traveling alone increases.

Let us focus now on the broadcast-time. Figure 4.13a shows a section A of size 4×4 of a grid R-SCS $\mathcal{F}$. Let S be the set of starting positions in A. Let $S' \subset S$ be the set of the starting positions marked as greater non-solid points. Notice that, from any point $p' \in S'$ we can reach a point $p \in S$ using a path of length 2π and all these paths traversed four communication links. Therefore, $M_{i,j} = \frac{1}{16}$ for all $i \in S'$ and $j \in S$ where M is the transition matrix of the discrete motion graph corresponding to $\mathcal{F}$. Notice that if $\mathcal{F}$ is big, most of the vertices in the discrete motion graph have degree 16 and the transition from these vertices to a neighbor has probability $\frac{1}{16}$. Therefore, if $\mathcal{F}$ is big the discrete motion graph is almost regular. In [34] the broadcast-time for regular graphs is studied and they state that for most starting positions, the expected time for k random walks to

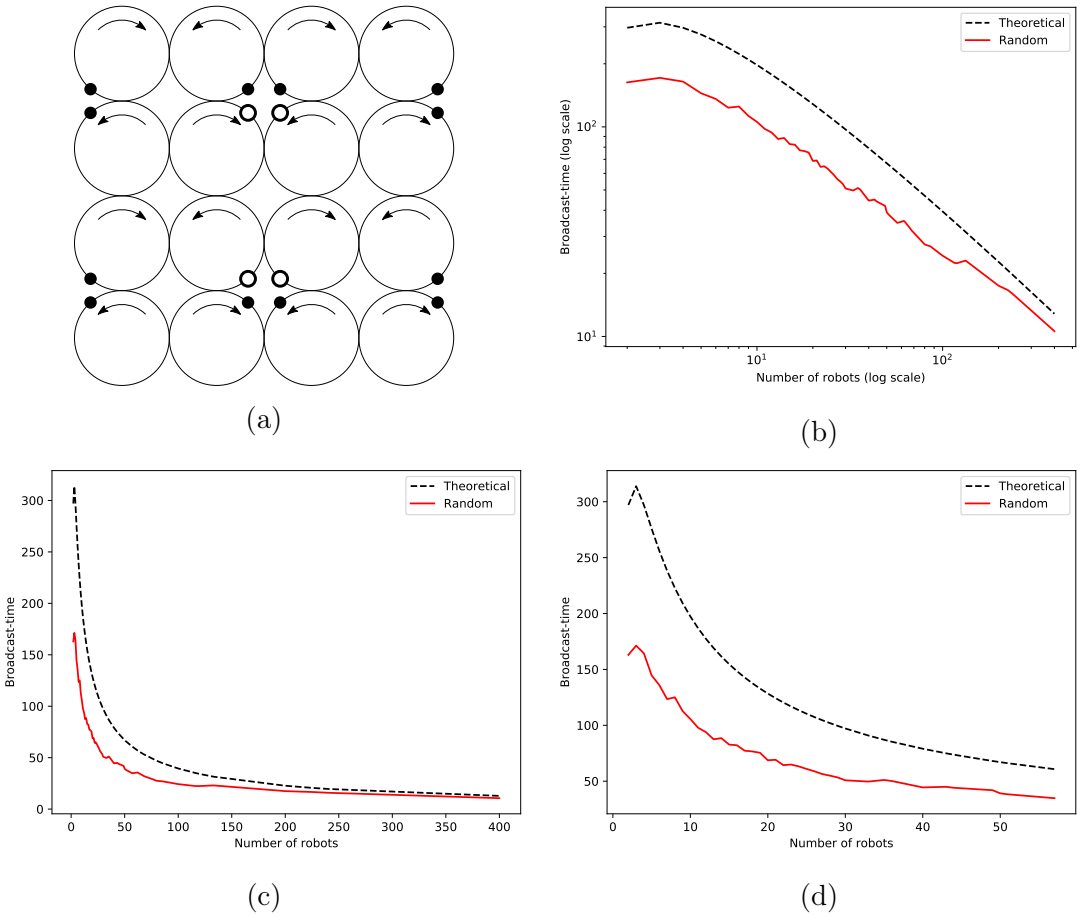


Figure 4.13: (a) A section of size 4×4 of a grid R-SCS. In (b), (c) and (d) the curve Random shows the average broadcast time obtained in several experiments in a 20×20 grid R-SCS using $k = 2...400$ robots. The curve Theoretical shows the behavior of the function $b(k) = \frac{2 \ln k(\delta-1)n}{k(\delta-2)}$ where $\delta = 16$ and $n = 400$. (b) Shows these curves using logarithmic scale, (c) shows these curves using linear scale and (d) shows this curves constrained to $k \leq 60$.

broadcast a single piece of information to each other is asymptotic to

$$\frac{2\ln k(\delta-1)n}{k(\delta-2)} \quad \text{as} \quad k,n\to\infty,$$

where n is the number of trajectories and δ is the vertices degree.

Figures 4.13 (b), (c) and (d) show a comparison between the average broadcast time results obtained in the experiments of subsection 4.3.4 using a 20×20 grid R-SCS $\mathcal{F}$ and the curve $b(k)=\frac{2\ln k(\delta-1)n}{k(\delta-2)}$ where $n=400$ (number of trajectories on $\mathcal{F}$) and $\delta=16$ (degree of most of the vertices in the discrete motion graph of $\mathcal{F}$). Analogously to the comparison for the isolation-time, the curve $b(k)$ is an upper bound of the real broadcast-time due to the same reason. Notice that the two curves have very similar shapes. In Figure 4.13 (d) these curves are shown for $k\leq 60$ in order to make visible the similarity on their shapes. Notice that both curves have a little peak at the beginning as we expect due to the natural behavior of the broadcast time when the number of robots increases (see final paragraph of subsection 4.3.4).

4.4 Conclusions

Terrain surveillance using cooperative unmanned aerial vehicles (UAV) with limited communication range can be modeled as a geometric graph in which the robots share information with their neighbors. If every pair of neighboring robots periodically meet at the communication link, the system is called a Synchronized Communication System (SCS). In order to add unpredictability to the deterministic framework proposed in Chapter 2, we study the use of stochastic strategies on the SCS. We evaluate both the coverage and communication performance and showed the validity of two random strategies compared with the deterministic one. The performance metrics of interest focused on in this chapter were *idle-time* (the expected time between two consecutive observations of any point of the system), *isolation-time* (the expected time that a robot is isolated) and *broadcast-time* (the expected time elapsed from the moment a robot emits a message until it is received by all the other robots of the team).

We first proved theoretical results for one of the strategies, the so-called random strategy, that can be modelled by classical random walks and obtain bounds assuming that the starting vertices are uniformly selected. A theoretical study for the other protocol, the so-called quasi-random strategy, that does not generate random walks, is a challenging open problem. Then, we performed three computational studies: a comparison between the deterministic and the random strategies; a simulation for estimating the mixing time of the system (roughly, the time needed for a random walk to reach its stationary distribution); and a comparison between the simulation and the theoretical bounds.

We state a summary of our observations in the experiments. Overall, the behavior was analyzed by increasing the size of the team of UAVs working on grid graphs. For the first study, our results pointed out that if we have few robots, as it is usual in practice, it is more convenient to use one of the random strategies instead of the deterministic one. Indeed, in this case, the results obtained with random strategies were much better than the behavior obtained using the deterministic strategy for the three tested measures. Although we can select clever initial positions for the robots in the deterministic strategy such that the properties of communication and coverage are satisfied, the system does not contemplate unpredictability. If some robots fail, the system can lose efficiency. As the experiments set, this drawback can be overcome using random strategies. The experiments also showed that the behavior of the random strategies is very similar for any of the proposed quality measures. In the second study, we observed that the estimated mixing time is small compared with the number of trajectories in the system and it increases slowly. The third experimental study showed that the theoretical bounds for the performance metrics are tight with the simulations.

Future research could also focus on considering other random strategies, different topologies in the experiments, as well as a study on the influence of the value of the probability p, fixed as 0.5 in this work.

Chapter 5

Block-sharing strategy

MULTI-ROBOT task allocation in dynamic environments is one of the fundamental problems in cooperative robotics and a challenging area in operational research [39, 52, 78, 126, 137]: given a group of cooperative robots, a global task to be performed in a dynamic environment and a cost function, how should subtasks be allocated to the robots in order to complete the task while minimizing costs?

Let us consider a task such as the cooperative manipulation of structures addressed in the ARCAS European Project (http://www.arcas-project.eu/) funded by the European Commission. One of the goals of this project is to assemble a structure using a team of aerial robots equipped with on-board manipulators. The practical interest of this system can be found in situations where it is required to build a structure in places with difficult access through conventional means (see Fig. 5.1). The use of aerial robots allows to perform assembly operations in any point in space, which in areas of difficult access represents a relevant advantage over ground robots.

Assembly planning [53] is the process of creating a detailed plan to craft a whole product from separate parts by taking into account the geometry of the final structure, available resources to manufacture the product, fixture design, feeder and tool descriptions, etc. Efficient assembly plans can significantly reduce time and costs. The assembly planning problem has been shown to be NP-complete [73] and covers three main assembly subproblems: sequence planning, line balancing, and path planning.

Reference [67] presents a classification of structures according to different features: number of hands, monotonicity (whether operations of intermediate placement of subassemblies are required), linearity (whether all assembly operations involve the insertion of a single part or multiple parts which have to be inserted

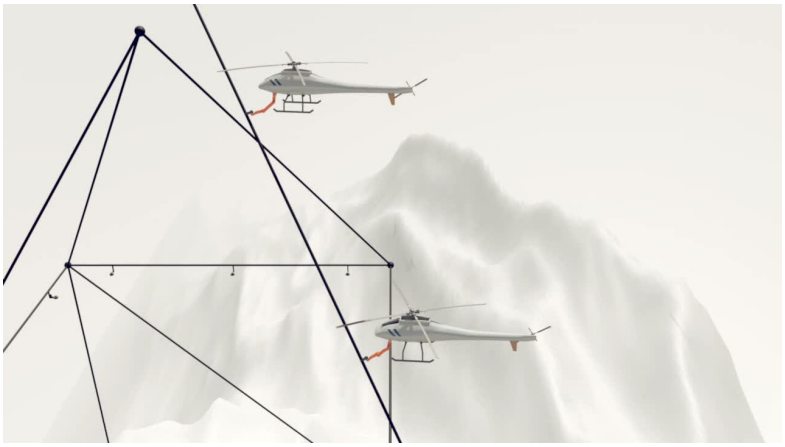


Figure 5.1: Two aerial robots equipped with LWR KUKA robotic arms manipulating a bar to build a structure in places with difficult access by conventional means.

simultaneously), and coherence (whether each part that is inserted will touch some other previously placed part). The structures considered in this chapter are sequential (for two robots), monotone, linear and contact-coherent.

This chapter focuses on the line balancing stage and introduces a novel paradigm for coordinating multiple robots in the execution of cooperative tasks in dynamic scenarios. The basic idea is to share information within a group or *block* of robots before assigning subtasks to all the members of the block. In this way, the method simulates a centralized system using a decentralized approach in which the robots share information in order to guarantee a local optimal solution in each stage of the algorithm. Our strategy fits the *locally centralized paradigm* mentioned in [27] and allows the team to perform task allocation periodically in an entirely decentralized manner. Moreover, by allocating tasks to robots repeatedly, a team can adapt as circumstances change achieving fluid coordination.

The block-information-sharing (BIS) strategy was introduced in [28] for area monitoring missions as a generalization of the one-to-one paradigm presented in [1, 2]. In these papers, the authors have experimentally demonstrated that this strategy converges to an optimal situation in which a particular objective function, the idle time function, is minimized. However, neither theoretical proofs on the convergence nor approach formalization have been published so far.

In this chapter, the convergence of the BIS strategy is formally proved in a general task allocation scenario. Moreover, it is shown how to use this strategy to design a fault-tolerant approach for structure construction using a cooperative team of aerial robots. The robots work in parallel and the dynamic assignments of the tasks are performed using blocks in order to maintain a balanced allocation where each aerial robot approximately spends the same time to construct the assigned section. Thus, the maximum time a robot spends to complete the assigned

section of the construction is minimized. Consequently, the benefit obtained is twofold: the total time for the construction is optimized and the robustness of the approach is guaranteed since the system is periodically re-balanced to cope with certain events, such as battery failure or the loss of some aerial robots, during the construction process.

5.1 Problem statement

Consider a cooperative team of autonomous and heterogeneous robotic agents and a task which has to be accomplished by the team. In a cooperative framework, it is assumed that the task is *divisible* and *parallelizable* in such a way that it can be partitioned so that each subtask is assigned to a robot in the team. Individual agents execute their subtasks independently except in the common parts where neighbors need to coordinate their cooperation.

The time to perform the overall task is determined by the maximum time spent by the individual robots on their subtasks while a balanced allocation needs to be maintained in order to prevent overload and ensure efficiency. Considering the parallel nature of this scenario, performing the task in the shortest possible time requires minimizing the maximum time spent by the individual robots.

As in the ALB problems, the task to perform can be modeled as a set of operations where subtasks correspond to a disjoint partition of the task. The formalization of the problem is illustrated with two examples:

(i) building a structure using assembly parts (we assume equal complexity of placing the parts) in which the task is the set of assembly operations;

(ii) monitoring a region, where the task is the set of all points in the region.

Let T be the set representing the general task to be performed, where each subtask is a subset of T. Let $U_1, U_2, \ldots, U_n$ be a team of n robots to perform the task. The *capability coefficient* indicates the amount of operations which can be performed by a robot in a given time frame. Let $c_i \in \mathbb{R}$ denote the capability coefficient of U_i to perform a task. For scenario (i), the capability coefficient of a robot is the number of assembly operations that can be performed per unit of time; for scenario (ii) it indicates the area which can be monitored in one unit of time. Let $\mu : \Sigma \to \mathbb{R}$ be a function to measure the task T, where Σ is a σ-algebra on T. For example, for a subtask $T' \subseteq T$, in (i) $\mu(T')$ is the number of assembly operations in T' and in (ii) $\mu(T')$ is the area of the region T'. The time required by the robot U_i to perform a subtask T' is given by

$$\frac{\mu(T')}{c_i}.$$

Let T_i be the subtask assigned to the robot U_i. Our goal is to obtain a partition $\{T_1, \ldots, T_n\}$ of T such that

$$\max_i \left\{ \frac{\mu(T_i)}{c_i} \right\}$$

is minimized. Using the definitions above, the key issue addressed in this chapter is the following:

Problem 5.1.1 (Efficient task allocation)**.** *Given a heterogeneous team of robots $U_1, U_2, \ldots, U_n$ and a task T,*

$$\begin{aligned} Minimize \quad & f(T_1, T_2, \ldots, T_n) = \max_i \left\{ \frac{\mu(T_i)}{c_i} \right\} \\ s.\ t. \quad & \bigcup_{i=1}^{n} T_i = T, \\ & T_i \cap T_j = \emptyset, \quad \forall i \neq j. \end{aligned}$$

If the general task T is a discrete set of atomic (indivisible, basic) operations, that is, $T = \{a_1, a_2, \ldots, a_m\}$, then for a subtask $T_i \subseteq T$, it follows that

$$\mu(T_i) = \sum_{a_j \in T_i} \mu(\{a_j\}) = \sum_{j=1}^{m} \mu(\{a_j\}) x_{ij}$$

where $x_{ij} = 1$ if $a_j \in T_i$, and $x_{ij} = 0$ otherwise. Our problem can be rewritten as follows:

$$Minimize \quad f(x_{11}, \ldots, x_{1m}, \ldots, x_{n1}, \ldots, x_{nm}) = \max_i \left\{ \frac{\sum_{j=1}^{m} \mu(\{a_j\}) x_{ij}}{c_i} \right\} \tag{5.1}$$

$$\begin{aligned} \text{s. t.} \quad & \sum_{i=1}^{n} \sum_{j=1}^{m} x_{ij} = m, \\ & \sum_{i=1}^{n} x_{ij} = 1, \quad \forall 1 \leq j \leq m. \end{aligned}$$

Note that a generalization of example (i), where the complexity of placing pieces varies and c_i is the capability of the i-th robot to manipulate pieces, can be also modelled using statement (5.1).

It is easy to see that in an optimal allocation all robots spend the same amount of time to perform their respective subtasks according to their capabilities. Consequently, to ensure an optimal task allocation, it follows that:

$$\mu(T_i) = c_i \frac{\mu(T)}{\sum_{j=1}^{n} c_j}. \tag{5.2}$$

In this chapter we are interested in a fault-tolerant and decentralized strategy to solve Problem 5.1.1 for multiple tasks. On the one hand, the conditions of the environment as well as the capabilities of the robots can change dynamically, requiring an update of the task allocation. On the other hand, in many real situations, robots work in an inaccessible areas and it is thus desirable to perform the tasks in a decentralized manner due to communication range limitations. Therefore, a decentralized strategy is required to recover and keep an optimal task allocation during the execution of a mission despite dynamical changes in the team or the environment. In the following section, the BIS strategy is presented as a solution to this problem.

In the remainder of this chapter, it is assumed that the global task to be performed can be represented as a measurable set T, and every subtask is a measurable subset T_i of T. Also, in order to simplify the notation, the values of $\mu(T)$ and $\mu(T_i)$ are referred to as T and T_i, respectively.

5.2 The block-information-sharing paradigm in dynamic task allocation problems

The so-called *one-to-one strategy* [1, 2], which was initially applied to area monitoring missions, can be easily adapted to general tasks. Suppose that our team has a non-optimal task allocation which we wish to turn into an optimal one in a decentralized manner without any intervention of external information sources. The key idea behind the one-to-one strategy is to share the information on capabilities and subtasks among neighboring robots so they can re-allocate subtasks between them. It has been experimentally demonstrated that the task allocation in the system converges to an optimal partition of the general task when this process is repeated iteratively. Unfortunately, the convergence may be too slow in some cases. In [28], examples of configurations where the one-to-one strategy converges only slowly to an optimal partition are shown and the BIS strategy is introduced as a generalization of the one-to-one approach that allows to accelerate

the convergence to an optimal solution. In this chapter, we lift the BIS strategy on a higher level of abstraction and present the approach as a general paradigm with multiple applications in cooperative robot systems.

5.2.1 General strategy

Consider a general task to be executed by a cooperative team of heterogeneous robots equipped with wireless communication for information sharing. Furthermore, assume that the environment conditions as well as the individual performance of the robots change dynamically affecting the global performance of the team. We are interested in a strategy to obtain and maintain a balanced subtask allocation which allows to complete the global task in minimum time. As the communication range is limited, it is assumed that each robot can only share information with its neighbors. This assumption implies a *communication graph*: Nodes correspond to robots and an edge between to nodes exists only if the respective robots are able to share information. To ensure that all members of the team are involved in the strategy, the communication graph needs to be connected. In Figure 5.2 a two-dimensional communication graph is illustrated (in real-world applications with aerial robots, a three-dimensional graph can be considered).

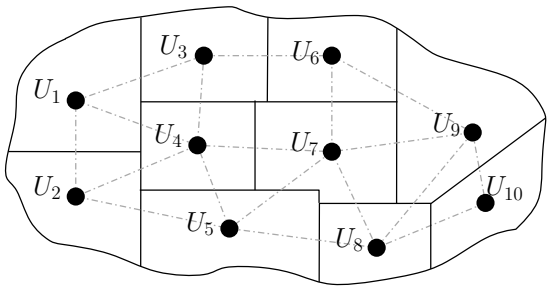


Figure 5.2: Representation of an area allocation among robots and the associated communication graph.

We define a *block* as a connected subgraph of the communication graph which we denote with the indices of the robots within the block. For example, $B = \{1, 3, 4\}$ denotes the block composed by the robots U_1, U_3 and U_4. A partition of the communication graph into blocks is a *block-configuration*. The BIS strategy supports two or more block-configurations such that their union contains all edges of the communication graph. Figure 5.3 shows two such block-configurations which, since their union contains all edges, can be used in the strategy.

The BIS strategy is based on the organization of a team of robots according to a block-configuration, where, within a block, robots share information among each other. Before creating an allocation, the robots remain in their position until all information about capabilities and tasks has reached all of the robots in the block. The iterative approach is as follows: When the team completes a task allocation

using a block-configuration, another block-configuration is used in the next stage. The reallocation process in a block uses equation (5.2) locally. Formally, if B is a block and U_i is a robot in B, the new subtask T_i' for U_i is given by

$$T_i' = c_i \frac{\sum_{j \in B} T_j}{\sum_{j \in B} c_j}. \tag{5.3}$$

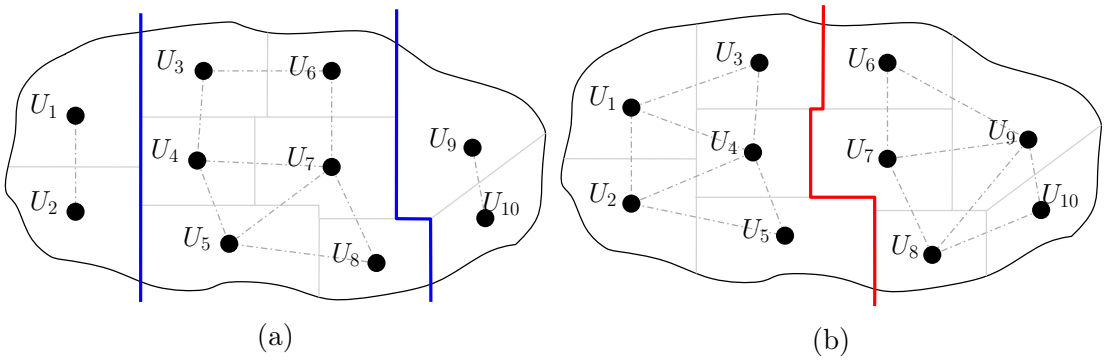


Figure 5.3: Block-information-sharing partition samples.

An example is given in Figure 5.3: Suppose that the robots start in the configuration shown in Figure 5.3a with partition into blocks $\{1, 2\}$, $\{3, 4, 5, 6, 7, 8\}$ and $\{9, 10\}$. Within each block, robots share information and reallocate their subtasks. After this first step, each block has a task allocation that is locally optimal. In the next stage, the team considers the configuration shown in Figure 5.3b composed of the blocks $\{1, 2, 3, 4, 5\}$ and $\{6, 7, 8, 9, 10\}$ and reallocates the tasks in these blocks correctly. After that, the team returns to the previous configuration in Figure 5.3a and so on.

In the next subsection, we formally prove that the BIS strategy, which takes advantage of local information to correctly distribute tasks, converges to an optimal task allocation for Problem 5.1.1. We aim to minimize the maximum time required by a robot to perform its subtask. In this scenario, which corresponds to an optimal solution, all heterogeneous robots spent an equal amount of time. The BIS strategy tends to produce such an allocation and ensures good performance for setups for which a balanced allocation is optimal. Consequently, this paradigm can be applied to multiple scenarios where a task is executed by cooperative robots. It allows to design efficient decentralized algorithms to optimize different objective functions, for instance, the maximum idle time in monitoring or surveillance missions [1, 2].

We furthermore identify the possibility of applying this scheme (sharing information and taking decisions in blocks while alternating block-configurations) in alternative scenarios where any type of information (which is not necessarily related to task allocation) is propagated in order to optimize the global system performance or a specific cost function (other than the robots spending an equal amount of time to perform their subtasks).

5.2.2 Convergence proof

The following conditions are assumed: the general task is divisible and parallelizable, the communication graph is connected and the union of the selected block-configurations is a covering of the set of edges of the communication graph.

The BIS strategy is an iterative approach where each iteration corresponds to a different block-configuration of the team. Let us represent by $T_r^{(j)}$ the subtask assigned to the robot U_r in the j-th iteration. If U_r belongs to block B in the j-th iteration then the equation (5.3) is written as

$$T_r^{(j)} = c_r \frac{\sum_{i \in B} T_i^{(j-1)}}{\sum_{i \in B} c_i}.$$

In the following results we use the values:

$$M^{(j)} = \max_i \left\{ \frac{T_i^{(j)}}{c_i} \right\} \qquad \text{and} \qquad m^{(j)} = \min_i \left\{ \frac{T_i^{(j)}}{c_i} \right\},$$

with $i \in \{1, \ldots, n\}$.

Lemma 5.2.1. *For all $k \geq 0$ it follows that*

$$m^{(j)} \leq \frac{T_i^{(j+k)}}{c_i} \leq M^{(j)} \quad \forall i \in \{1, \ldots, n\}.$$

Proof. We proceed to use mathematical induction in k. If $k = 0$, then $m^{(j)} \leq \frac{T_i^{(j+k)}}{c_i} = \frac{T_i^{(j)}}{c_i} \leq M^{(j)}$ by definition of $M^{(j)}$ and $m^{(j)}$. Assume as an induction hypothesis that for a fixed value k the claim is fulfilled, $m^{(j)} \leq \frac{T_i^{(j+k)}}{c_i} \leq M^{(j)}$. Let U_r be an arbitrary robot and let B be the block where U_r lies in the $(j+k+1)$-th iteration, then:

$$T_r^{(j+k+1)} = c_r \frac{\sum_{i \in B} T_i^{(j+k)}}{\sum_{i \in B} c_i}.$$

From the hypothesis we deduce:

$$\begin{aligned}
c_i m^{(j)} \leq & \quad T_i^{(j+k)} & \leq c_i M^{(j)} \\
\textstyle\sum_{i\in B} c_i m^{(j)} \leq & \quad \textstyle\sum_{i\in B} T_i^{(j+k)} & \leq \textstyle\sum_{i\in B} c_i M^{(j)} \\
m^{(j)} \cdot \textstyle\sum_{i\in B} c_i \leq & \quad \textstyle\sum_{i\in B} T_i^{(j+k)} & \leq M^{(j)} \cdot \textstyle\sum_{i\in B} c_i \\
m^{(j)} \leq & \quad \dfrac{\sum_{i\in B} T_i^{(j+k)}}{\sum_{i\in B} c_i} & \leq M^{(j)} \\
c_r m^{(j)} \leq & \quad c_r \dfrac{\sum_{i\in B} T_i^{(j+k)}}{\sum_{i\in B} c_i} & \leq c_r M^{(j)} \\
c_r m^{(j)} \leq & \quad T_r^{(j+k+1)} & \leq c_r M^{(j)}
\end{aligned}$$

and the result follows. □

Lemma 5.2.2. *For all $k, l \geq 0$ it follows that:*

$$\begin{aligned}
&\textit{(a) if } m^{(j)} < \frac{T_i^{(j+k)}}{c_i} \quad \textit{then } m^{(j)} < \frac{T_i^{(j+k+l)}}{c_i} \\
&\textit{(b) if } \frac{T_i^{(j+k)}}{c_i} < M^{(j)} \quad \textit{then } \frac{T_i^{(j+k+l)}}{c_i} < M^{(j)}
\end{aligned}$$

Proof. Only claim (a) is proven since the proof for claim (b) is analogous. Mathematical induction in l will be used. If $l = 0$, it follows that

$$\frac{T_i^{(j+k+l)}}{c_i} = \frac{T_i^{(j+k)}}{c_i} > m^{(j)}.$$

Suppose as induction hypothesis that $m^{(j)} < \dfrac{T_i^{(j+k+l)}}{c_i}$ for a fixed value l. Let U_r be an arbitrary robot and let B be the block where U_r lies in the $(j+k+l+1)$-th iteration, then:

$$\begin{aligned}
T_r^{(j+k+l+1)} = c_r \frac{\sum_{i\in B} T_i^{(j+k+l)}}{\sum_{i\in B} c_i} > c_r \frac{\sum_{i\in B} c_i m^{(j)}}{\sum_{i\in B} c_i} = \\
= c_r \frac{m^{(j)} \sum_{i\in B} c_i}{\sum_{i\in B} c_i} = c_r m^{(j)}
\end{aligned}$$

and the result follows. □

We are ready to prove the main result of this section.

Theorem 5.2.3. *The block-information-sharing strategy always converges to an optimal task allocation for Problem 5.1.1.*

Proof. Let $T_i^{(*)}$ be the task assigned to U_i in an optimal task allocation. Since the time required for every subtask in an optimal task allocation is the same, it is enough to prove that the differences between the times converge to 0. Let $\rho^{(j)}$ be the maximum time difference between the performances of every pair of robots in the j-th iteration,

$$\rho^{(j)} = \max_{i,l} \left\{ \left| \frac{T_i^{(j)}}{c_i} - \frac{T_l^{(j)}}{c_l} \right| \right\} = M^{(j)} - m^{(j)}. \tag{5.4}$$

Note that $\rho^{(j)} \geq 0$ for all j and that the maximum time difference between the performance of every pair of robots in the optimal partition is 0 because all the robots spend the same amount of time to perform their subtasks. We prove that $\rho^{(j)}$ is a decreasing function and then

$$\lim_{j \to \infty} \rho^{(j)} = 0.$$

Suppose that $\rho^{(j)} = c > 0$, we prove that there exists a $q > 0$ such that $\rho^{(j+q)} < \rho^{(j)}$. Let G be the communication graph of the team. Let U_r and U_p be two adjacent nodes in G such that $\frac{T_r^{(j)}}{c_r} = M^{(j)}$ and $\frac{T_p^{(j)}}{c_p} < M^{(j)}$. These two nodes exist since otherwise the graph G would not be connected or $\frac{T_i^{(j)}}{c_i} = M^{(j)}$ for all the nodes and then $\rho^{(j)} = 0$. The union of all the block-configurations contains all the edges of G, so the edge (U_r, U_p) is in some block of a block-configuration. Let $(j+k)$ with $k > 0$ be the index of the first iteration in which the team takes this configuration and U_r and U_p lie in the same block B. Since $\frac{T_p^{(j)}}{c_p} < M^{(j)}$, by Lemma 5.2.2 we have

$$\frac{T_p^{(j+k-1)}}{c_p} < M^{(j)}. \tag{5.5}$$

Also, from Lemma 5.2.1 follows $\frac{T_i^{(j+k-1)}}{c_i} \leq M^{(j)}$ for all $i \in \{1, \ldots, n\}$, especially for all $i \in B$, then

$$\sum_{i \in B} T_i^{(j+k-1)} \leq \sum_{i \in B} c_i M^{(j)}. \tag{5.6}$$

From $p \in B$, using (5.5) and (5.6), follows that:

$$\begin{aligned} \sum_{i \in B} T_i^{(j+k-1)} &< \sum_{i \in B} c_i M^{(j)}, \text{ then dividing by } \sum_{i \in B} c_i \\ \frac{\sum_{i \in B} T_i^{(j+k-1)}}{\sum_{i \in B} c_i} &< \frac{\sum_{i \in B} c_i M^{(j)}}{\sum_{i \in B} c_i} = M^{(j)}. \end{aligned} \tag{5.7}$$

Multiplying (5.7) by c_b with $b \in B$ and using equation (5.3) we conclude that

$$T_b^{(j+k)} = c_b \frac{\sum_{i \in B} T_i^{(j+k-1)}}{\sum_{i \in B} c_i} < c_b M^{(j)}.$$

Since $r \in B$ then $T_r^{(j+k)} = c_r \frac{\sum_{i \in B} T_i^{(j+k-1)}}{\sum_{i \in B} c_i} < c_r M^{(j)}$, that is, $\frac{T_r^{(j+k)}}{c_r} < M^{(j)}$. Using the same argument for every robot U_i having $\frac{T_i^{(j)}}{c_i} = M^{(j)}$ we induce that there exists a value $k_M > 0$ such that $\frac{T_i^{(j+k_M)}}{c_i} < M^{(j)}$ for all i.

Analogously we can prove that there exists a value $k_m > 0$ such that $m^{(j)} < \frac{T_i^{(j+k_m)}}{c_i}$ for all i.

Therefore, there exists $q > 0$ ($q = \max\{k_m, k_M\}$) such that for every robot U_i in the team we have $m^{(j)} < \frac{T_i^{(j+q)}}{c_i} < M^{(j)}$, and $\rho^{(j+q)} = M^{(j+q)} - m^{(j+q)} < M^{(j)} - m^{(j)} = \rho^{(j)}$, proving the theorem.

□

5.3 A case study: structure construction

In this section the block-information-sharing paradigm is applied to the construction of a 3D structure using a team of cooperative robots. In line with the objectives of the ARCAS project[1], which considers scenarios involving damaged infrastructure, we have chosen the example of a bridge: The structure is composed of different assembling parts (bars) which must be assembled in a specific order attending to restrictions of union, gravity, etc. (precedence constraints), see Figure 5.4. The basic component of the bridge is a cube and the length of a bridge is given by the number of cubes composing it, see Figure 5.4.

The parts to be assembled have been deployed (i.e. air-dropped) in specific locations which are referred to as *stores*. Robots should transport parts taken from the stores to the structure site and assemble them correctly. In this scenario, it is more convenient to use aerial robots to avoid obstacles on the ground (also while the structure grows it is becoming an obstacle to the movement of the ground robots). Furthermore, if the structure is multi-level, one bearing another, it is easier for an aerial robot to place part on part in a stack. The aerial robots are equipped with on-board manipulators for the assembly operations.

With the start of the mission, a subtask is assigned to each aerial robot (AR) according to its capabilities. During the mission, a variety of incidents including difficulties while assembling a part, battery degradation, and motor or dexterous manipulator failure, may delay an AR's schedule. In this case, the AR may have to

[1] http://www.arcas-project.eu

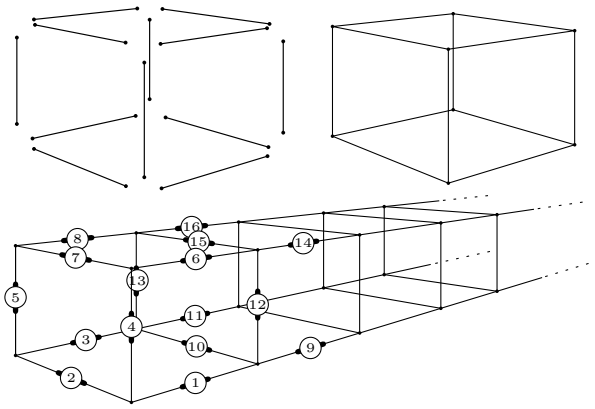


Figure 5.4: Bridge structure. The numbers indicate a possible enumeration of the parts.

abandon the mission and its subtasks must be reallocated to ensure the completion of the task.

5.3.1 The Problem

Consider a team of ARs equipped with dexterous manipulators to assemble a bridge (one level) as shown in Figure 5.4. Each AR has a wireless communication device that allows the information interchange with its neighbors and each assembly part has an embedded radio transmitter for its localization and identification. The required sequence of assembly operations has been computed based on assembly-by-disassembly techniques [67, 114] in advance. The assembly tasks should be divided into subtasks which can be executed in a parallel manner, to avoid waiting for the placement of all the supporting parts. The robots should also give higher priority to the common parts in order to minimize the waiting time for their neighbors. According to Problem 5.1.1, the goal is to keep an assembly operations allocation among all available ARs in such a way that the maximum time an AR spends to execute its subtasks is minimized in a fault-tolerant manner, that is, considering that the number of robots and their capabilities can dynamically change.

5.3.2 The strategy

A bridge is an elongated structure which can be conveniently divided into longitudinal sections, see Figure 5.5. This partition generates separated workspaces in which ARs can operate in parallel with low collision risk. When the partition is given, a number of assembly parts is assigned to each AR according to its capabilities. From this partition, the team can be represented by the sequence $U_1, U_2, \ldots, U_n$ where U_i is the AR corresponding to the i-th section of the bridge.

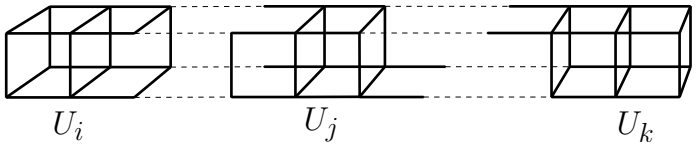


Figure 5.5: The bridge is divided into sections assigned to each AR of the team.

We proceed to the design of a decentralized algorithm based on the BIS strategy. First, a sorted list of bridge parts $P = \{p_1, p_2, \ldots, p_m\}$ is computed considering dependencies between them and their positions in the structure from left to right (details of a possible ordering are given in Section 5.3.4). Every continuous subsequence of P determines a longitudinal section of the bridge (see Figure 5.5). Note that if the subsequence $p_j, p_{j+1}, \ldots, p_{j+r}$ corresponds to U_i, then the subsequence corresponds to U_{i+1} is $p_{j+r+1}, \ldots, p_{j+r+s}$.

Since the workspaces are adjacent sections of the bridge, every robot, except the robots working at the end points, has two neighbors. In Figure 5.6a, three robots are shown: U_1 and U_2 are sharing information, and U_2 is able to communicate with U_3 if they move closer to each other. In this scenario, the communication graph in the team of ARs is a simple line as shown in Figure 5.6b.

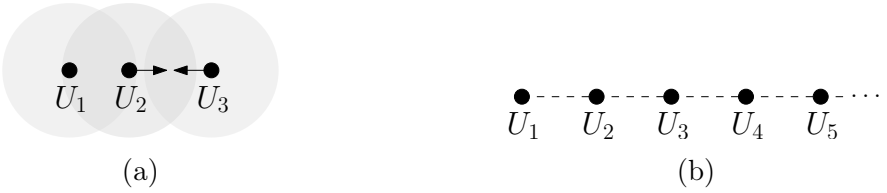


Figure 5.6: The communication graph in a line.

A block in this graph is a continuous subsequence of nodes (ARs) composing a line segment. Therefore, a block-configuration corresponds to a list of consecutive segments. We will work with two block-configurations where all the blocks have the same size except the first or last blocks. Figure 5.7a shows two block-configurations with block size two, one in gray and the other in white. Note that these block-configurations cover the set of edges entirely and can be used in a BIS strategy to redistribute the workload among the team. Figure 5.7b shows a similar case with blocks of size four.

The BIS strategy in this setting works as follows: In certain time intervals, the members of a block share information on placed and unplaced pieces and on their capabilities among them. Once all robots in a block receive this information, a sorted list of unplaced pieces is computed and then a new allocation according to the current capabilities of the available ARs (within the block) is considered. Subsequently, the members of the team change according to the new block-configuration.

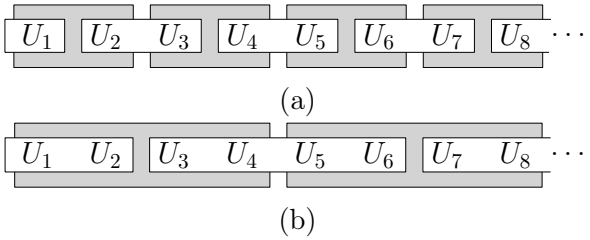


Figure 5.7: Block-configurations with blocks of length two (a) and four (b), respectively.

To illustrate the strategy, consider the following example: Let $U_1, U_2, \ldots, U_7$ be a team of seven ARs having the same capability of normalized value one. Each team member starts with 20 assembly parts and the time spent by an AR (with capability one) to find and place an assembly part is 5u (five units of time), see Figure 5.8. Suppose that, after 20u, the capability of U_4 has decreased to 0.5 and it has assembled 3 pieces while the others have assembled 4 pieces, see Figure 5.8. With the new capability, U_4 spends 10u to find and place an assembly part (twice the time spent by others). If the team works without sharing information, the time to complete the structure is the time spent by U_4 to perform its subtask, $20\text{u} + 17 * 10\text{u} = 190\text{u}$. We now demonstrate that this can be improved when the BIS strategy is applied: We assume that the team is using two block-configurations with blocks of size four (see Figure 5.7b). The team alternates the block-configurations starting with the gray one. Assume the ARs use an additional 2u to share information and allocate the assembly operations into the block properly. We furthermore assume that team members share information in periodic intervals of 20u. In the interval 20u-22u the team makes a division using the gray block-configuration and in the interval 22u-42u each AR places four parts except U_4 which assembles two parts. After that, in the interval 42u-44u, the ARs update the balanced allocation of the assembly operations applying the white block-configuration and so on. Following this strategy (with no further changes in the team), it becomes obvious that the team completes the structure in 120u. Note that the less time spent on information exchange, the better the overall performance.

Let us now analyze a case where individual ARs fail. Obviously, if the team does not share information, the structure will not be completed if one or more ARs fail. To address this problem in the BIS strategy, the team proceeds as follows: When an AR does not meet a neighbor then it moves in the same direction to meet a new neighbor. When such a meeting occurs (or the AR reaches the end of the bridge), the ARs which should be between the meeting ARs are considered missing. In this way, the remaining robots obtain the new graph representation and assume the tasks of the failed robots. For instance, in Figure 5.9 the robot U_{i+2} has failed. When U_{i+1} and U_{i+3} try to meet the right and left neighbors, respectively, they meet each other and consider U_{i+2} as missing. In the example

time	U_1	U_2	U_3	U_4	U_5	U_6	U_7
0u	20/1	20/1	20/1	20/1	20/1	20/1	20/1
20u	16/1	16/1	16/1	17/0.5	16/1	16/1	16/1
22u	19/1	19/1	19/1	8/0.5	16/1	16/1	16/1
42u	15/1	15/1	15/1	6/0.5	12/1	12/1	12/1
44u	15/1	15/1	13/1	6/0.5	13/1	13/1	12/1
64u	11/1	11/1	9/1	4/0.5	9/1	9/1	8/1
66u	10/1	10/1	10/1	5/0.5	9/1	9/1	8/1
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮

Figure 5.8: Example of the evolution of allocated tasks of the team $U_1, U_2, \ldots, U_7$ using the BIS strategy with blocks of size four. Each cell contains information on the robot in the column's header at the time indicated by the row's header. The format of the cells is p/c, where p is the number of pieces assigned to the robot and c is the capability coefficient.

of Figure 5.9, if the failure of U_{i+2} is detected using the gray block-configuration, then the other ARs in the block given by U_i, U_{i+1} and U_{i+3} resume the unfinished assembly operations of U_{i+2} in the reallocating process of the task and continue the alternating process between the gray and white configurations.

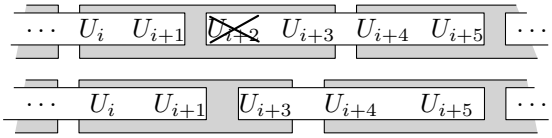


Figure 5.9: The block-information-sharing strategy with a failure in U_{i+2}.

Unfortunately, the system cannot always recover so easily from failures. Let us consider the same situation mentioned above, but now the AR U_{i+3} also fails. In this new situation, if the ARs try to keep the same block-configuration, the edge (U_{i+1}, U_{i+4}) will be not present in any block (see Figure 5.10) and the convergence to an optimal task allocation is not guaranteed.

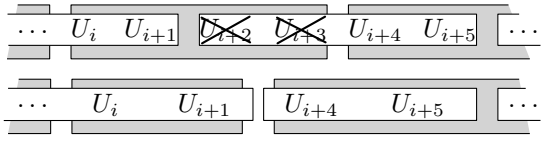


Figure 5.10: Block-sharing method with failures in U_{i+2} and U_{i+3}.

A generalization of this situation is used to explain how to solve this problem: Assume a team of ARs working with regular block-configurations with blocks of

size $2k$. Figure 5.11 depicts this generalization: each dot-dashed rectangle with a k inside represents a sequence of k ARs and consequently, two consecutive rectangles form a block of size $2k$. The problem appears when k consecutive ARs fail at the extreme of a block, for example, the failure of the third sequence of k ARs in Figure 5.11a. We can solve this issue by modifying the block-configuration in such a way that every connection edge between two neighbors is covered. Figure 5.11b shows the recovered system. Notice that the configuration remains unchanged to the right of the bold vertical line whereas it is changed (inverted) to the left. This reconfiguration can be applied the other way around, keeping the left side and reversing the right side. In order to minimize the time to obtain a recovered system in the propagation process, the shortest side should be modified. The new block-configuration can be locally computed with a simple data structure on each agent.

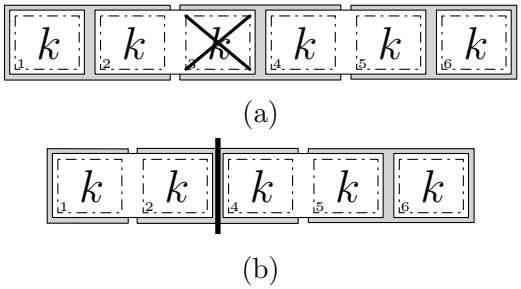


Figure 5.11: Recovering of the BIS strategy using $2k$ block size when k consecutive ARs are missing.

Finally, in case new robots join the team, it is assumed that they enter at the endpoints of the communication line. With this assumption, the ARs can manage the additions easily, making the respective updates in the blocks containing the extremes at which the addition occurs.

5.3.3 Implementation

In this section we provide pseudo-code of two algorithms running on-board each AR in parallel. One of them addresses the processing of the assembly operations and the other one handles information sharing and reallocations of the assembly operations into the corresponding blocks.

Algorithm 40 shows the steps to process the assigned list of assembly operations to be performed. Each AR executes the same algorithm with local information. The variables (in italic font), functions and procedures (in math font) used in the algorithm are explained subsequently:

- $P^{(i)}$ is the list of assembly parts assigned to the AR U_i (local task to be performed by U_i).

- S is a description of the whole structure to be built; each part is unique and corresponds to a specific place in S.
- *searching* indicates if the AR is searching for a part.
- *carrying* indicates if the AR is carrying a part with its robotic arm.
- p is the target part to pick or place.
- *sharing* indicates if the robot's communication interface is open (this value is managed by Algorithm 37).
- `found`(p) returns **true** if the robot is over the part p.
- `load`(p) activates the robotic arm and picks up part p.
- `plan_to_put`(S, p) elaborates and loads a flight plan to carry part p and place it correctly into structure S.
- `continue`() follows the current flight plan.
- `correct_place`(S, p) returns **true** if the AR has reached the correct position to place part p into S.
- `put`(p) places the part p in its position in S.
- `next_piece`($S, P^{(i)}$) selects the next piece of $P^{(i)}$ to assemble in S according to the dependency rules and priority of the common parts preventing neighbors from waiting.
- `search`(p) uses the radio signal of p to locate it and elaborates a flight plan to reach it.
- `release`(p) is a protocol to release the part p in case of failure or other cases of mission cancellation.

Algorithm 37 shows the logical steps to share information and reallocate the assembly operations inside the blocks. It uses methods that encapsulate the ideas exposed in the previous section.

- T is the time interval to share information; the AR must share information every T units of time (this value is equal for all the ARs).
- t is the duration of the sharing process, $t < T$.
- Pp is the set of known placed parts.

Algorithm 5: On-board algorithm to process the assigned list of assembly parts.

```
   Input  : P^(i), S
 1 searching ← false
 2 carrying ← false
 3 p ← none
 4 while not ABORT do
 5     if not sharing then
 6         if searching then
 7             if found(p) then
 8                 load(p)
 9                 searching ← false
10                 carrying ← true
11                 plan_to_put(S, p)
12             else
13                 continue()
14             end
15         else
16             if carrying then
17                 if correct_place(S, p) then
18                     put(S, p)
19                     p ← next_piece(S, P^(i))
20                     if p is not none then
21                         searching ← true
22                         search(p)
23                     end
24                     carrying ← false
25                 else
26                     continue()
27                 end
28             else
29                 p ← next_piece(S, P^(i))
30                 if p is not none then
31                     searching ← true
32                     search(p)
33                 end
34             end
35         end
36     end
37 end
38 if ABORT and carrying then
39     release(p)
40 end
```

- Ids is the list of identifiers of all ARs in the team. $Ids[j]$ corresponds to the identifier of U_j and is used to determine if an AR is new in the team or if any AR has failed.
- BS stores a description of the block-configurations to use in the sharing process, including block size and start positions of the blocks. It allows to determine the members composing the block at any given time .
- *sharing* indicates if the communication interface of the robot is open.
- b indicates the current block-configuration in BS.
- Q_e stores a list of detected events in the team: failures and incorporations.
- K stores the knowledge of the ARs in the team (placed and unplaced parts, and current capabilities of the team members).
- `time_to_share`(T, t) returns **true** if it is time to share information.
- `current_capabilities`() estimates its own current capabilities using on-board sensors and time spent performing the last operations.
- `prepare_info`$(K, Pp, P^{(i)}, c_i)$ prepares the local information for sharing (current knowledge, known placed parts, unplaced parts in plan and its own current capabilities).
- `open_connections`() activates and opens the communication interface (to save energy, the communication interface only activated during sharing).
- `time_to_close`(T, t) returns **true** if it is time to close the communication interface.
- `close_connections`() turns the communication interface off if the AR is not sharing information.
- `block`(BS, b, Ids) elaborates a representation of the block to perform a re-allocation of the assembly operations.
- `pieces_in_block`(B, K, S) computes the set of unplaced parts to assign in the block.
- `capabilities_in_block`(B, K) returns the list of the capabilities of the available members in the block.
- `division`(P, C) makes a division between the available members in the block according to their capabilities.

- `update_team`(Q_e, Ids) updates the list of available ARs and the block-configurations according to detected events (failures and incorporations).
- `next_conf`(b, BS) returns the next block-configuration to use (remember that the block-configurations are alternating).
- `meet_neighbor`() connects with a neighbor if possible.
- `graph_event`(U, Ids) detects whether an event has occurred: If the identifier of U is not present in Ids, then U is a new AR incorporated into the team. If U is in Ids but the predecessor or successor is not, then the members between them have failed. If U is **none** then all members from the current position to the bridge end are considered failed.
- `gen_event`(U, Ids) returns a representation of the occurred event to put it in the list of events.
- `share`(U, Q_e, K) sends all the known information to U and returns the information detected by U.
- `merge_info`$(info, Q_e, K, Ids)$ updates the detected information (Q_e and K) using the information received from U.

A video available at `https://www.youtube.com/watch?v=BshZ9tSQt9I` illustrates the evolution of the task allocation algorithm in a team using these algorithms.

5.3.4 Simulations and computational results

In this subsection we provide a detailed description of the implementation of algorithms 40 and 37 for a simulation of the case study presented above using the BIS strategy. We furthermore analyze the effect of the block size on efficiency and robustness and compare the results with state-of-the-art algorithms.

Parts assignment

In a first step, all parts are ordered so that a subsequence of the sorted list of pieces corresponds to a section of the bridge. The pieces have been numerated following the scheme of Figure 5.12 from left to right, see also Figure 5.4.

Subsequently, a precedence graph of three levels is constructed. The first level describes bottom parts, the second one vertical parts and the third one top parts. In each level, the pieces are sorted using the assigned index. The precedence graph displaying the numbered parts of the bridge of Figure 5.4 is shown in Figure 5.13. The idea behind this particular order is to be able to process the pieces in the resulting graph from left to right without violations of the dependency rules. In the

Algorithm 6: Decentralized algorithm to keep a distribution of the assembly parts according to the capabilities of the team members.

```
   Input : T, t, P^(i), S, Ids, BS
 1 sharing ← false
 2 b ← 0
 3 Q_e ← [ ]
 4 K ← {}
 5 while not ABORT do
 6     if not sharing then
 7         if time_to_share(T, t) then
 8             c_i ← current_capabilities()
 9             K ← prepare_info(K, Pp, P^(i), c_i)
10             open_connections()
11             sharing ← true
12         end
13     else
14         if time_to_close(T, t) then
15             close_connections()
16             B ← block(BS, b, Ids)
17             P ← pieces_in_block(B, K, S)
18             C ← capabilities_in_block(B, K)
19             d ← division(P, C)
20             P^(i) ← d[i]
21             [Ids, BS] ← update_team(Q_e, Ids)
22             b ← next_conf(b, BS)
23             sharing ← false
24         else
25             U ← meet_neighbor()
26             if graph_event(U, Ids) then
27                 e ← gen_event(U, Ids)
28                 Q_e ← Q_e ∪ {e}
29             end
30             info ← none
31             if U is not none then
32                 info ← share(U, Q_e, K)
33             end
34             [Q_e, K] ← merge_knowledge(info, Q_e, K, Ids)
35         end
36     end
37 end
```

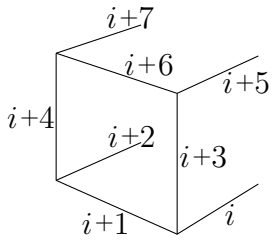


Figure 5.12: Enumeration of bridge's parts.

example shown in Figure 5.13, the ordering is $1, 2, 4, 3, 5, 7, 9, 12, 6, 10, 11, 13, 8, \ldots$. Note that every subsequence of this ordering corresponds to a section of the bridge. Furthermore, in a sharing stage, the set of unplaced pieces is ordered using the positions of the pieces in this total ordering and then is divided according to the capabilities of every robot in the block in order to obtain a workspace division in the new task allocation. Note that through working in this manner, the parts assigned to each robot in the block belong to disjoint sections in the bridge.

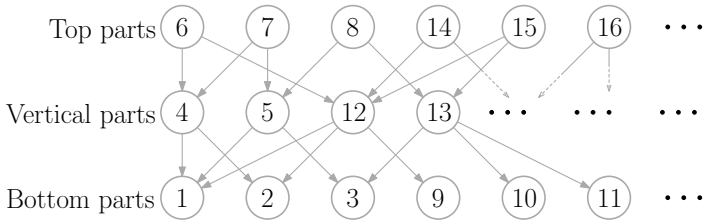


Figure 5.13: Precedence graph of the bridge's parts.

Simulation architecture

The architecture of our simulation is shown in Figure 5.14: The robots are equipped with an implementation of the previously presented algorithms. The *environment* entity simulates all sensor-related processes of the robots. It manages the positions of the robots and pieces, the state of the pieces and takes control of simulation rules.

There are three states for pieces in a simulation:

- *available* (the piece is available to be taken by a robot),
- *carrying* (the piece has been loaded by a robot and it is moving towards the structure), and
- *placed* (state reached when the robot carrying the piece has found the correct position of this piece in the structure and placed it).

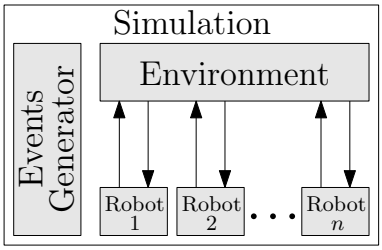


Figure 5.14: Simulation architecture.

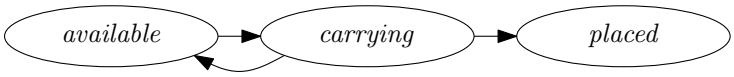


Figure 5.15: Considered states of the pieces and the transitions between them.

Figure 5.15 shows the transitions between these states. A piece passes from *carrying* state to *available* only when the carrying robot leaves the team.

We define six states for the ARs:

- *idle* (when the simulation starts, or when it finishes the current list of assigned parts),
- *searching_piece* (the robot is searching for a piece to place),
- *picking_piece* (the robot found the searched piece and is picking it up),
- *carrying_piece* (the robot picked up the target piece and is carrying it to the bridge location),
- *placing_piece* (the robot found the final position of the piece and is placing it, after that, the robot decides the next piece to place and passes to the *searching_piece* state) and
- *searching_neighbor* (this state is reached from the states *searching_piece* and *placing_piece* when the time frame to meet a neighbor has passed).

Figure 5.16 shows the transitions between robot states: If a robot is in *placing_piece* state and has completed the list of assigned assembly operations, it passes to *idle* state. When a robot is in *searching_neighbor* state and meets a neighbor or reaches the bridge end, it proceeds as follows: if there are assembly operations to perform, it determines the next piece to place and passes to *searching_piece* state, but if there are no assembly operations to perform (in its assigned list), it passes to *idle* state.

In our simulation, we introduce the *skill coefficient* of a robot as a value to model its ability to manipulate pieces. Higher values of ability correspond to lower required time to lift or place a piece. A robot with a robotic arm with 6

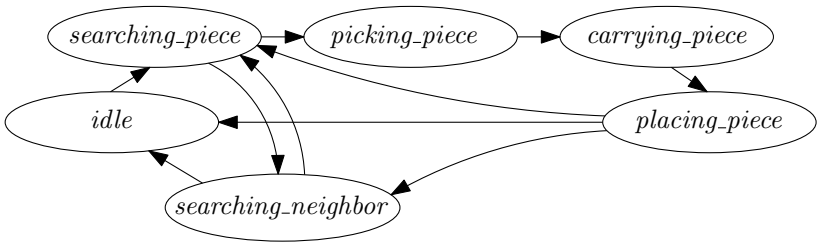


Figure 5.16: The states of each robot and the transitions between them.

degrees of freedom has a higher skill coefficient than a robot with 4 degrees of freedom. Also, this coefficient can be interpreted as the operability state of the robotic arm (a robotic arm with mechanical problems has a skill coefficient lower than a robot in perfect working conditions).

The *environment* manages the transitions between the states. For instance, a robot can only lift up a piece if the piece is available and the robot is flying over the piece. It furthermore controls the advance and current state of the lifting and placing operations according to the current skill coefficient of the robot, and keeps track of the positions of the robots according to their motion directions and speeds. Note, that in this case the capability coefficient of a robot is a combination of speed and skill.

The other important entity in our simulation is the *events generator*, which generates random events of mission abandonment, incorporation of members and changes on the capabilities (speed or skill coefficient) of a specific robot.

Scenario description

We consider for the simulation a scenario where the bridge to be built is composed of bars with a length of one meter. All pieces have been dropped in a single place (store) at a distance of 20 meters from the bridge location. The communication range of the robots is 8.0m, their initial speed ranges from 3.7m/s to 4.0m/s and their initial skill coefficients are set between 2.9 and 3.0 (randomly distributed following a uniform distribution). We are assuming that the needed effort to lift and assemble a piece is 18 and 90, respectively. Thus, a robot with skill coefficient 3.0 spends $18/3.0 = 6$s to lift a piece and $90/3.0 = 30$s to place it in the structure. Note that the measure of the effort to place a single piece correctly is a combination of the effort to lift it, the required effort to transport it to the final location in the structure and the effort to place it.

Figure 5.17 shows the different states of a team of eight robots during a simulation. At the beginning, robots are located above the structure location (Figure 5.17a) and have an initial assembly assignment. After assessing the first required piece, the robots head towards the store (Figure 5.17b and Figure 5.17c). Note that various robots can be at the store simultaneously. In our simulation, the

store is represented as a single point and we assume that the robots have implemented a local strategy of collision avoidance. In real-word applications the store is usually a region.

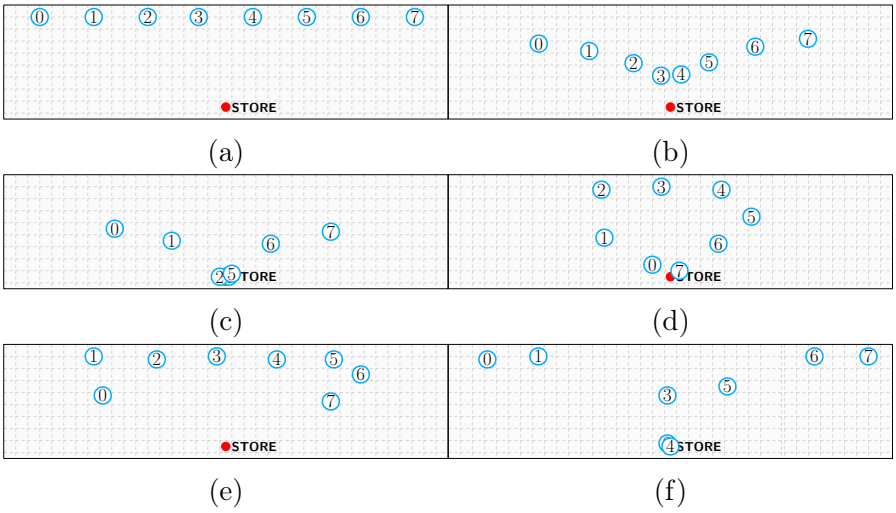


Figure 5.17: Motion of the robots between the structure and the store.

Figures 5.17d and 5.17e show the robots on their way to the structure location carrying the pieces to be placed. Note that the robots in the middle reach the structure first and the robots closest to the bridge ends last. In Figure 5.17e it can be seen that, while robot 3 is assembling a piece, robot 0 is still carrying a piece to the correct place in the structure. In this simulation, the sharing process is not marked by time intervals since a robot executes the local sharing process when it has new information of all the members in the current block. To avoid endless waits, a timeout for information sharing takes place when a member in the block fails. Figure 5.17f shows an arbitrary instant illustrating how robots move independently from the store to the structure, execute their assembly operations and exchange information when connected with a neighbor near the structure (the pair of robots $(0, 1)$ and $(6, 7)$ may be sharing information). The video `https://youtu.be/0m6J1cqO1PI` shows the initial stages of a simulation.

Results with different strategies

Two possible information sharing procedures can be applied to the scenario applied above, a local or a global information retrieval before the decision process. So far, related approaches, the *one-to-one* and the *centralized* strategy, represent strict versions of these choices. We compare such algorithms to the BIS strategy introduced in this chapter. In the one-to-one strategy, the information is shared between only two neighboring robots. This approach has been used frequently in related literature [1, 3, 2]. On the other end of the spectrum are centralized

algorithm with global knowledge, where a new allocation is created based on all the information available in the entire system. However, the BIS strategy is decentralized and takes advantage of quick local decisions using a group of agents to achieve global solutions. Note that the block size in the BIS strategy indicates the degree of global knowledge in the reallocation process. Furthermore the three algorithms can be integrated within the BIS strategy, namely, size 2 for one-to-one and size n for the centralized approach, where n is the number of robots in the team.

Using blocks of big size implies a more centralized approach which allows the computation of a near optimal solution since global knowledge is taken into account. However, collecting, processing, and broadcasting all globally available knowledge can be time consuming which limits the efficiency of the overall system. Additionally, an almost centralized approach may not yield improvement: If some agents in a large block leave the team, the task allocation cannot be completed until the failure has been detected by all other block members. In contrast, a more decentralized approach using small blocks achieves robustness to individual failures and may gain a performance advantage through parallel computation. However, the quality of the solution may decrease. In this section we explore this trade-off between efficiency and robustness.

In order to analyze the balance of the task allocation in the team during a mission we introduce some notations. Let $P_i(t)$ and $c_i(t)$ be the number of unplaced pieces (workload) assigned to the i-th robot and its capability coefficient at instant t of the mission, respectively. From equation (5.2) it follows that in a team with n robots at instant t, the task allocation is balanced if:

$$\frac{P_j(t)}{c_j(t)} = \frac{\sum_{i=1}^{n} P_i(t)}{\sum_{i=1}^{n} c_i(t)}, \text{ for all } 1 \leq j \leq n. \tag{5.8}$$

Let $r_i(t) = P_i(t)/c_i(t)$ be the ratio of workload and capability of the i-th robot at instant t and let $r^*(t) = \sum_{i=1}^{n} P_i(t) / \sum_{i=1}^{n} c_i(t)$ be the optimum ratio of workload and capability at time instant t according to (5.8).

In the scope of this study, we consider the following function to assess the imbalance of the workload in the system:

$$\xi(t) = \sqrt{\frac{\sum_{i=1}^{n} (r_i(t) - r^*(t))^2}{n}}. \tag{5.9}$$

Note that if the workload in the team is balanced then the value of ξ is zero, while a higher value of ξ corresponds to a greater degree of imbalance.

Figure 5.18 depicts the comparison among different strategies for a setup with 8 robots and a bridge of 100 cubes length. The tested strategies are: one-to-

one, centralized and BIS strategy. The latter was applied in two regular block-configurations with blocks of size 4 and 6, respectively. We will denote by *O-O*, *CENT* and *BIS-k* the strategies one-to-one, centralized and BIS strategy with block size k, respectively. The results indicate that CENT is the most time consuming. The team spends 7554.5s with O-O, 7446.5s with BIS-4, 7433.5s with BIS-6 and 8594.0s with CENT. During the simulation, the *events generator* produces two random fail-events (consequently information sharing is needed to finish the construction of the bridge).

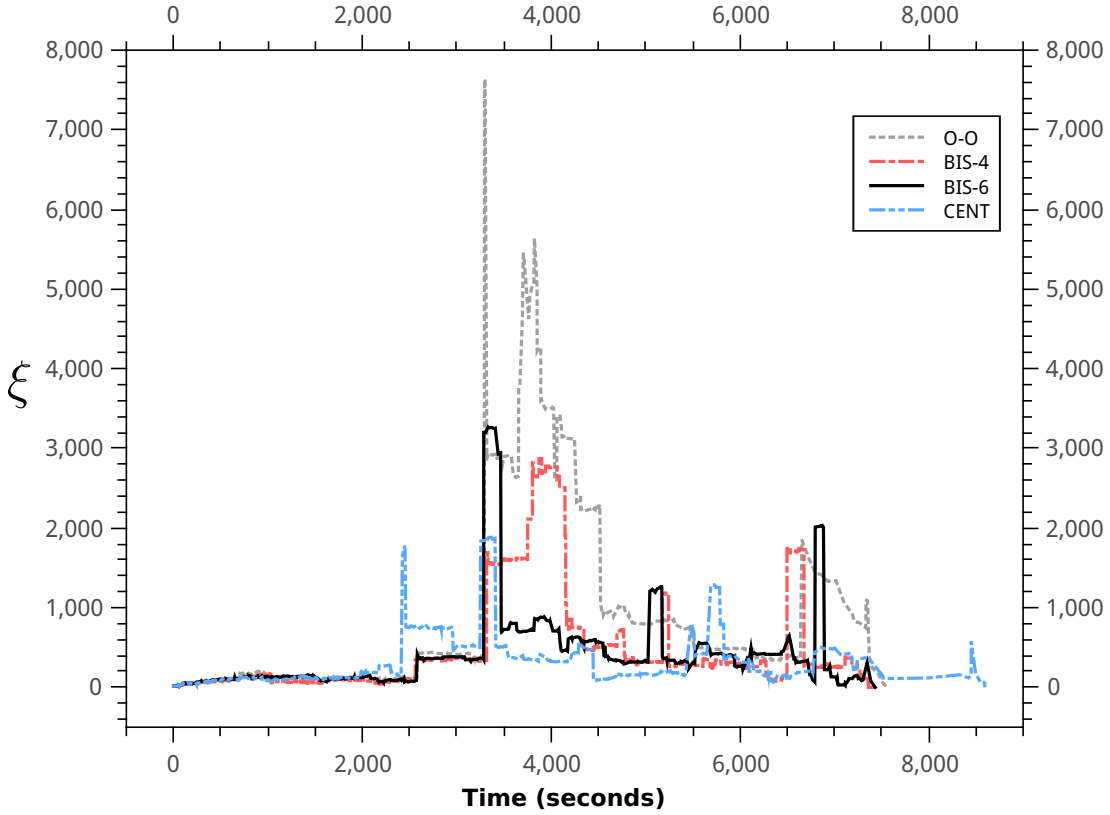


Figure 5.18: Behavior of the function ξ in a simulation using different strategies: the one-to-one strategy (O-O, dashed gray line); BIS strategy with block size 4 (BIS-4, dash-dotted red line); BIS strategy with block size 6 (BIS-6, continuous black line) and centralized strategy (CENT, dash-dot-dotted blue line).

In Figure 5.18, every peak in a curve represents a disturbance in the workload balance due to a sharing process. Initially, the team starts with an optimal task allocation and thus, at the beginning of the simulation, the values of ξ are close to 0. Around second 2400 of the simulation, we observe an abrupt increase of imbalance for the team using the centralized strategy. The cause of this behavior is the high volume of information which needs to be collected. The members of the team

spend a relatively large amount of time retrieving information from all teammates. In fact, when the leftmost robot receives the information of the rightmost robot, the information is already outdated. Suppose the identifiers of the robots from left to right are $r_1, r_2, \ldots, r_8$ and robot r_8 shares information with its neighbor r_7. At this moment, r_7 receives the information about the number of unplaced pieces assigned to r_8. At a certain instant in time, r_7 passes this information to r_6, and so on until this information reaches r_1. While the information about unplaced pieces assigned to r_8 is traveling to reach r_1, r_8 continues working. Therefore, at the moment in which r_1 receives the information about r_8, the number of unplaced pieces assigned to r_8 that receives r_1 is greater than the actual number of unplaced pieces assigned to r_8. This causes an alteration in the task allocation, assigning some placed pieces to the robots as unplaced pieces.

Around instant 3250s of the simulation, an event occurs which produces an imbalanced workload (note that all the curves have a strong disturbance). The curve corresponding to the one-to-one strategy experiences stronger disruption than the others due to the following reason: When a robot detects that its neighbor has failed, it resumes the assembly task of its neighbor and its workload grows significantly with respect to other workloads. However, using blocks of robots (with sizes greater than 2) in the reallocation, the assembly task is divided between the remaining members of the block thus obtaining better distribution (this argument also applies to the centralized strategy).

Larger blocks imply fewer reallocation operations in order to obtain an optimal task allocation. Therefore, in most cases faster convergence to an optimal partition is guaranteed. Note in Figure 5.18 that the team using one-to-one spends more time to reach a balanced task allocation from a peak than others. On the other hand, if the changes in the system occur faster than the broadcast time in the block, the convergence to an optimal task allocation may be put on hold. As an example, see the behavior shown in Figure 5.18 around second 2400 of the simulation using the centralized strategy. Also, using the BIS strategy with larger blocks (or a centralized strategy) may cause larger time intervals between reallocations. This may result in a problem if for instance a robot u is trying to place a piece which requires another piece assigned to a neighbor u', and u' fails. Then u will be locked trying to place its piece until the next reallocation occurs. Another undesirable situation occurs if a robot u is idle (with no assigned pieces because it has finished its assigned assembly operations) while its neighbor u' still has pending operations. In this case, u will be idle and waiting for a new assignment until the next reallocation process occurs. Both situations may increase the execution time of the global task as illustrated in Figure 5.18 by the curve corresponding to the centralized strategy.

In summary, in this specific experiment, BIS-6 seems to be the most adequate choice to maintain a balance between efficiency and robustness.

Finally, in order to analyze both, workload variability and total time needed to

assemble the structure, for the chosen strategy, we performed several experiments using a team starting with 10 robots. Different bridge lengths were tested: 40, 60, 80 and 100 cubes. For each length, we performed 50 runs using different strategies: O-O, BIS-4, BIS-6, BIS-8 and CENT. We have implemented the scenario introduced in subsection 5.3.4 and the occurrence of failures in the system has been generated randomly.

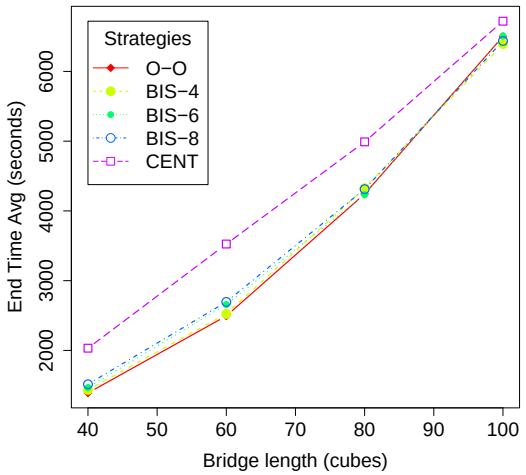
Tables 5.1a, 5.1b, 5.1c and 5.1d show the obtained results for each bridge length. We have considered three parameters: the total time to assemble the bridge (*End Time*) in seconds, the mean value of function ξ (*mean-*ξ) and the maximum value of function ξ (*max-*ξ). The tables show the average, best and worst cases (*Avg*, *Best* and *Worst*, respectively) for each parameter. The best and worst average value of these parameters have been highlighted in the tables. Figure 5.19 illustrates the growth of the parameters using the average data.

The results (Tables 5.1a-5.1d) indicate that the centralized strategy is the worst among the considered methods: it obtained the lowest efficiency and furthermore ranked low with respect to balancing. The other strategies appear to have a similar behavior regarding efficiency, whereas BIS shows a better performance with respect to balancing compared to O-O.

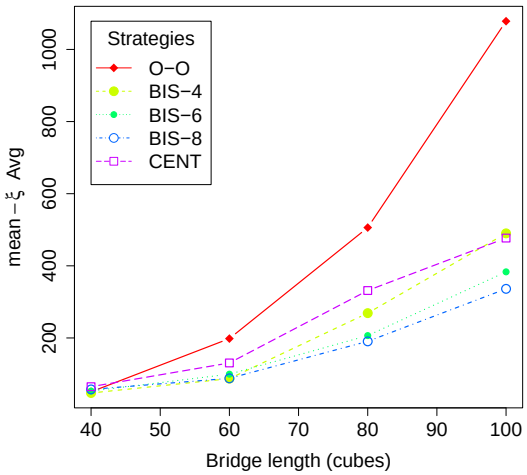
5.4 Conclusions and future developments

In this chapter we have introduced the block-information-sharing strategy as a new paradigm for performing task allocation in a decentralized manner. This strategy is applicable to a variety of scenarios meeting the following conditions: the general task should be divisible and parallelizable, the communication graph should be connected and the union of the selected block-configurations should be a covering of the set of edges of the communication graph. It has been proved that if these conditions are fulfilled, the convergence to an optimal task allocation can be guaranteed. Thus, the experimental proofs on the convergence in related literature [1, 3, 2, 28] are now theoretically endorsed.

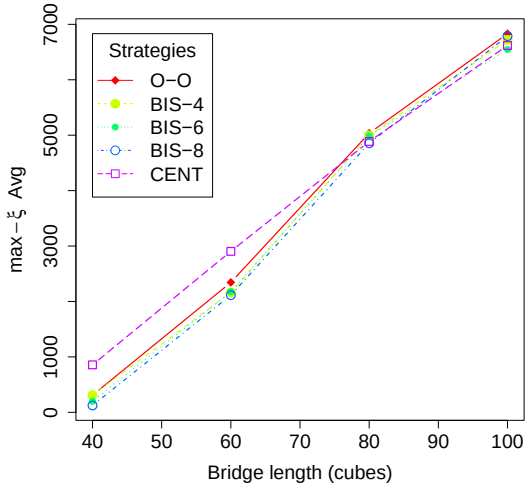
It has also been demonstrated that the block size is an important parameter in this strategy which is directly related to the rate of convergence and robustness. In an ideal scenario with instantaneous communication and no limit in the information amount to be handled by one robot, the convergence rate increases with increasing block size. However, in real applications, sharing with blocks of big size may be expensive and the system can lose in efficiency and fault tolerance. In fact, the memory of the robots is limited and time is spent while moving to ensure the information sharing process. Thus, although the benefits of the BIS strategy are visible in the computational experiments in both, surveillance tasks [28] and assembling structures, the block size is a parameter that can be adequately set based on prior studies and simulations for a given scenario. Indeed, the lack of a priori knowledge of the block size for a specific communication graph is the bot-



(a)



(b)



(c)

Figure 5.19: Performance of End Time (a), mean-ξ (b) and max-ξ (c).

tleneck of the BIS paradigm. In this work, the block size is considered an input parameter, whereas the computation of an optimal value for a given configuration remains an open problem.

	End Time (seconds)			mean-ξ			max-ξ		
	Avg	Best	Worst	Avg	Best	Worst	Avg	Best	Worst
O-O	**1389.53**	1325.00	1636.00	50.59	29.94	237.32	308.73	56.18	2666.22
BIS-4	1434.30	1311.00	2248.00	**47.39**	38.91	67.14	309.49	65.28	3008.07
BIS-6	1467.51	1315.50	1978.00	52.74	41.49	61.92	199.44	79.32	2827.56
BIS-8	1514.89	1322.50	3344.50	57.36	27.35	71.07	**123.82**	89.77	853.34
CENT	**2031.71**	1348.00	6587.50	**64.84**	43.11	86.07	**856.64**	94.70	2826.23

(a)

	End Time (seconds)			mean-ξ			max-ξ		
	Avg	Best	Worst	Avg	Best	Worst	Avg	Best	Worst
O-O	**2494.09**	2222.50	3243.00	**198.21**	42.81	866.00	2344.04	72.68	4554.40
BIS-4	2523.77	2189.50	3351.50	**87.18**	46.38	226.73	2162.54	88.97	4180.97
BIS-6	2658.44	2222.00	4675.00	99.58	63.37	304.14	2176.64	114.84	5140.51
BIS-8	2694.95	2203.00	3937.00	88.02	56.37	153.99	**2114.48**	127.39	4647.83
CENT	**3525.19**	2256.00	4722.50	130.69	47.08	411.49	**2904.37**	105.25	4979.16

(b)

	End Time (seconds)			mean-ξ			max-ξ		
	Avg	Best	Worst	Avg	Best	Worst	Avg	Best	Worst
O-O	4243.14	3523.50	6783.00	**506.05**	66.79	1376.72	**5035.26**	347.14	9060.06
BIS-4	4326.19	3469.00	6131.50	268.84	59.96	806.60	4994.50	112.68	7387.02
BIS-6	**4232.80**	3496.00	7645.50	206.53	77.42	595.44	4987.09	131.83	7424.13
BIS-8	4311.82	3501.00	6269.50	**190.56**	91.45	603.06	**4854.98**	169.49	7510.42
CENT	**4989.96**	3479.00	7415.5	331.48	62.51	827.66	4886.42	119.46	6950.91

(c)

	End Time (seconds)			mean-ξ			max-ξ		
	Avg	Best	Worst	Avg	Best	Worst	Avg	Best	Worst
O-O	6500.33	5656.50	8622.00	**1078.28**	112.09	2518.39	**6831.81**	205.28	10282.73
BIS-4	**6392.32**	5494.00	8276.50	490.07	80.61	1181.09	6717.91	142.26	10380.09
BIS-6	6509.25	5564.00	8850.50	383.30	101.29	864.76	**6546.49**	174.50	9953.94
BIS-8	6436.99	5532.00	8896.00	**336.21**	119.54	876.99	6787.17	452.30	8624.41
CENT	**6719.93**	5740.00	8982.50	476.96	130.54	757.68	6618.10	217.68	9027.23

(d)

Table 5.1: Summary of *end time*, *mean* value of function ξ and *maximum* value of function ξ using different strategies for a setup with 10 robots and different bridge lengths. The average, best and worst value of each of these parameters is shown. The tables (a), (b), (c) and (d) corresponds to bridges of 40, 60, 80 and 100 cubes length, respectively.

Chapter 6

Summary and future work

Motivated by the growing interest on problems that appear in aerial robotics systems and its applications, which often are not studied from a formal and general point of view, this thesis addressed several challenges on aerial multi-robot systems from a discrete and combinatorial optimization perspective. The presented work has been developed in the framework of several research projects focused on aerial robotics systems and, it is centered in the theoretical analysis of some problems related to these projects, rather than focusing on the engineering part related to the hardware, implementation and commissioning of these systems.

The goal of this thesis, presented as an applied mathematical work, is two-fold. One is to solve robotics problems using mathematical resources and, another one is to post theoretical concepts and problems, inspired by aerial robotics applications, that could be of great interest to solve for mathematicians. More specific objectives of the thesis are: (1) the synchronization problem, which asks for a coordination strategy that allows periodical communication between the members of a cooperative team while performing a task in a scenario with limited communication range. (2) To study the robustness of the strategy proposed as a solution for the synchronization problem, it must be able to recover in case of catastrophic failures of some robots. And, (3) online (periodical) task allocation in a cooperative team of aerial robots with limited communication range.

Chapter 2 is dedicated to the synchronization problem. According to our methodology, it starts by formally posting the problem and, after that, a simplification by using unit circular trajectories is considered. Using this simple model, we make a rigorous study of the problem and we present *synchronized systems* as a solution. In this chapter, we present a characterization of the optimal solutions and propose an algorithm to find them. We also show how to extend our results in the simplified model to more general scenarios. In order to address the sec-

ond of the specific objectives of the thesis, we introduce the *shifting strategy* as a deterministic protocol to recover the system when some robots leave. Finally, in order to validate our proposals, we include the results of several computational simulations and a preliminary real experiment. Future research lines in this area include the study of other variations of the proposed problem considering non-simple trajectories and allowing more than one communication link between two trajectories. Also, it could be interesting to study recovery strategies to deal with the inability of an agent to properly maintain its schedule along its trajectory.

Although the robustness of a synchronized system was initially addressed in Chapter 2, it is more deeply analyzed in Chapters 3 and 4. The computational simulations at the end of Chapter 2 show a good performance of the shifting strategy when a small set of drones leave the system. However, it is shown in Chapter 3 that in the worst cases, the leaving of some robots causes that some trajectory points are no longer visited, or some robots never meet another one, or the system loses the ability to relay messages between any pair of robots. Chapter 3 contains the most theoretical part of the thesis. It starts mathematically defining a *synchronized communication system (SCS)* as a dynamic structure where the robots move following some strict rules. Then, combinatorial problems on the resilience of a SCS are stated: *What is the minimum number of robot that must fail such that*

1. *some trajectory point is not longer visited?*

2. *k of the surviving robots will never meet another robot to share information?*

3. *the system is not able to relay a message between some pair of surviving robots?*

These problems could be interesting for engineers because they allow to measure the robustness of a system: the greater the resilience values are, the most robust the system is. However, due to the problem statement itself and, the aesthetic beauty of the structures and techniques to solve them, these problems are also attractive for mathematicians. Throughout this chapter we showed connections of these problems with number theory, graph theory, circulant graphs and polynomial multiplication. From the theoretical point of view, as future research, we could study other properties (not necessarily related with robustness) of the synchronized systems. For example, at the end of Chapter 3, the design of a more specific algorithm to compute the vertex-connectivity of the graph that models the communications between the robots, is suggested as a future research line.

Chapter 4 introduces some stochastic behavior for the drones in order to avoid the drawbacks mentioned above. Also, some measures, commonly used in the literature, are considered to evaluate the performance of a system. These measures are:

- *idle-time*: the average time that trajectory points remain unobserved by a robot,

- *isolation-time*: the average time that robots spend without communicating with any other robot, and,

- *broadcast-time*: the average time elapsed from the moment a robot emits a message until it is received by all robots.

Two random strategies are proposed as alternatives for the deterministic recovery strategy introduced in Chapter 2. One of these strategies is easier to implement in practical scenarios because the robots fly in separate regions. The other one allows that two or more robots fly together in the same region, thus, it requires some protocol to avoid collisions. However, this last strategy is easier to study theoretically because it generates a classical random walk for each active robot. A section of Chapter 4 is dedicated to this study and we prove some bounds for the expected values of the measures stated above. Moreover, several statistical experiments were performed showing that the two random strategies have a very similar performance and are much better than the deterministic one, specially if the number of robots in the system is small with respect to the number of trajectories. Future research could focus on a study on the influence of the decision probability factor p (fixed as 0.5 in this work) in different topologies of the system communication graph.

In Chapter 5, the *block-information-sharing (BIS)* strategy is formally presented as a generalization of the *one-to-one* strategy introduced in [2, 1]. Having a team of robots with limited communication range, which are operating in a large scenario where is not possible to maintain a permanent communication, the BIS strategy addresses the problem of maintain a balanced task allocation among the robots by using intermittent and periodical communications. We have formally proven that the it converges to an optimal task allocation in an abstract and general scenario. Additionally, we have shown how to use this approach to design a fault-tolerant decentralized algorithm for structure construction using a cooperative team of aerial robots.

The BIS strategy and the synchronized communication systems with random behavior showed very promising results with respect to performance and robustness. However, they require a lot of engineering work to be implemented in real systems, in fact, several issues related to sensor precision, error management and communication should be addressed to implement these solutions on reliable systems. Notice that, although these strategies emerged from aerial multi-robot applications, they could be extended and generalized to any kind of multi-robot system (e.g., ground robots, underwater robots, etc.). Also, the proposed strategies could be useful to develop other variants in specific scenarios or may inspire new heuristics.

Lastly, we would like to mention that, we have recently been involved in a new project consisting on filming a sport event with a team of aerial robots. The main underlying problem of this project is: given a filming plan consisting of several scenes (a scene is a filming location or a continuous sequence of filming locations with an associated time window), schedule a flight plan for every available robot such that all the scenes are filmed. Some variants of this problem have been studied before by using directed graphs, however, the main novelty in our scenario is that the drones have a limited battery and they need to periodically return to a charging station. This is another promising scenario for future works in the interplay between mathematics and robotics.

Bibliography

[1] J.J. Acevedo, B.C. Arrue, J.M. Díaz-Báñez, I. Ventura, I. Maza, and A. Ollero. Decentralized strategy to ensure information propagation in area monitoring missions with a team of UAVs under limited communications. In *Unmanned Aircraft Systems (ICUAS), 2013 International Conference on*, pages 565–574, 2013.

[2] J.J. Acevedo, B.C. Arrue, J.M. Díaz-Báñez, I. Ventura, I. Maza, and A. Ollero. One-to-one coordination algorithm for decentralized area partition in surveillance missions with a team of aerial robots. *Journal of Intelligent and Robotic Systems*, 74(1-2):269–285, 2014.

[3] J.J. Acevedo, B.C. Arrue, I. Maza, and A. Ollero. Distributed approach for coverage and patrolling missions with a team of heterogeneous aerial robots under communication constraints. *International Journal of Advanced Robotic Systems*, 10(28):1–13, 2013.

[4] M. Adbelhafiz, A. Mostafa, and A. Girard. Vehicle Routing Problem Instances: Application to Multi-UAV Mission Planning. In *AIAA Guidance, Navigation, and Control Conference*, 2010.

[5] N. Agatz, P. Bouman, and M. Schmidt. Optimization Approaches for the Traveling Salesman Problem with Drone. *Transportation Science*, 52(4):965–981, 2018.

[6] A. Albert and L. Imsland. Combined Optimal Control and Combinatorial Optimization for Searching and Tracking Using an Unmanned Aerial Vehicle. *Journal of Intelligent & Robotic Systems*, 95(2):691–706, 2019.

[7] D. Alejo, J.A. Cobano, G. Heredia, J.R. Martínez-de Dios, and A. Ollero. Efficient trajectory planning for WSN data collection with multiple UAVs. In *Cooperative Robots and Sensor Networks 2015*, pages 53–75. Springer, 2015.

[8] A. Almeida, G. Ramalho, H. Santana, P. Tedesco, T. Menezes, V. Corruble, and Y. Chevaleyre. Recent advances on multi-agent patrolling. In *Brazilian Symposium on Artificial Intelligence*, pages 474–483. Springer, 2004.

[9] M. Alzenad, A. El-Keyi, F. Lagum, and H. Yanikomeroglu. 3-D placement of an unmanned aerial vehicle base station (UAV-BS) for energy-efficient maximal coverage. *IEEE Wireless Communications Letters*, 6(4):434–437, 2017.

[10] M. Alzenad, A. El-Keyi, and H. Yanikomeroglu. 3-D placement of an unmanned aerial vehicle base station for maximum coverage of users with different QoS requirements. *IEEE Wireless Communications Letters*, 7(1):38–41, 2017.

[11] C. Avin and B. Krishnamachari. The power of choice in random walks: An empirical study. *Computer Networks*, 52(1):44–60, 2008.

[12] D. Bayer and P. Diaconis. Trailing the Dovetail Shuffle to its Lair. *Annals of Applied Probability*, 2(2):294–313, 1992.

[13] I. Bekmezci, O.K. Sahingoz, and Ş. Temel. Flying ad-hoc networks (FANETs): A survey. *Ad Hoc Networks*, 11(3):1254–1270, 2013.

[14] T. Bektas. The multiple traveling salesman problem: an overview of formulations and solution procedures. *Omega*, 34(3):209 – 219, 2006.

[15] J. Bellingham, M. Tillerson, A. Richards, and J.P. How. *Multi-Task Allocation and Path Planning for Cooperating UAVs*, pages 23–41. Springer US, Boston, MA, 2003.

[16] J. L. Bentley, D. F. Stanat, and E. H. Williams Jr. The complexity of finding fixed-radius near neighbors. *Information Processing Letters*, 6(6):209 – 212, 1977.

[17] J.J. Bentley. Fast Algorithms for Geometric Traveling Salesman Problems. *ORSA Journal on Computing*, 4(4):387–411, 1992.

[18] M. Bernard, K. Kondak, and G. Hommel. Load transportation system based on autonomous small size helicopters. *Aeronautical Journal*, 114(1153):191–198, 2010.

[19] L. F. Bertuccelli, H.L. Choi, P. Cho, and J. P. How. Real-time multi-UAV task assignment in dynamic and uncertain environments. In *AIAA Guidance, Navigation, and Control Conference*, 2009.

[20] F. Boesch and R. Tindell. Circulants and their connectivities. *Journal of Graph Theory*, 8(4):487–499, 1984.

[21] N. Boysen, M. Fliedner, and A. Scholl. A classification of assembly line balancing problems. *European Journal of Operational Research*, 183(2):674–693, 2007.

[22] L. Brunet, H.L. Choi, and J. P. How. Consensus-based auction approaches for decentralized task assignment. In *AIAA Guidance, Navigation and Control Conference and Exhibit*, 2008.

[23] A. P. Brunner. Isolation in synchronized drone formations. Master's thesis, University of Denver, 2015.

[24] P. Bupe, R. Haddad, and F. Rios-Gutierrez. Relief and emergency communication network based on an autonomous decentralized uav clustering network. In *SoutheastCon 2015*, pages 1–8. IEEE, 2015.

[25] W. Burgard, M. Moors, C. Stachniss, and F. E. Schneider. Coordinated multi-robot exploration. *Robotics, IEEE Transactions on*, 21(3):376–386, 2005.

[26] T. Camp, J. Boleng, and V. Davies. A survey of mobility models for ad hoc network research. *Wireless communications and mobile computing*, 2(5):483–502, 2002.

[27] Y. U. Cao, A. S. Fukunaga, and A. Kahng. Cooperative mobile robotics: Antecedents and directions. *Autonomous robots*, 4(1):7–27, 1997.

[28] L.E. Caraballo, J.J. Acevedo, J.M. Díaz-Báñez, B.C. Arrue, I. Maza, and A. Ollero. The block-sharing strategy for area monitoring missions using a decentralized multi-uav system. In *Unmanned Aircraft Systems (ICUAS), 2014 International Conference on*, pages 602–610. IEEE, May 2014.

[29] D. W. Casbeer, D. B. Kingston, R. W. Beard, and T. W. McLain. Cooperative forest fire surveillance using a team of small unmanned air vehicles. *International Journal of Systems Science*, 37(6):351–360, 2006.

[30] H. Choi, L. Brunet, and J. P. How. Consensus-based decentralized auctions for robust task allocation. *IEEE Transactions on Robotics*, 25(4):912–926, 2009.

[31] C. M. Clark, S. M. Rock, and J. C. Latombe. Motion planning for multiple mobile robots using dynamic networks. In *Robotics and Automation, 2003. Proceedings. ICRA'03. IEEE International Conference on*, volume 3, pages 4222–4227. IEEE, 2003.

[32] B. Codenotti, I. Gerace, and S. Vigna. Hardness results and spectral techniques for combinatorial problems on circulant graphs. *Linear Algebra and its Applications*, 285(1):123 – 142, 1998.

[33] A. Collins, J. Czyzowicz, L. Gasieniec, A. Kosowski, E. Kranakis, D. Krizanc, R. Martin, and O. Morales Ponce. Optimal patrolling of fragmented boundaries. In *Proceedings of the twenty-fifth annual ACM symposium on Parallelism in algorithms and architectures*, pages 241–250. ACM, 2013.

[34] C. Cooper, A. Frieze, and T. Radzik. Multiple random walks in random regular graphs. *SIAM Journal on Discrete Mathematics*, 23(4):1738–1761, 2010.

[35] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein. *Introduction to Algorithms*. The MIT Press, 3rd edition, 2009.

[36] J. Czyzowicz, L. Gasieniec, A. Kosowski, and E. Kranakis. Boundary patrolling by mobile agents with distinct maximal speeds. In *European Symposium on Algorithms*, pages 701–712. Springer, 2011.

[37] J. Czyzowicz, A. Kosowski, E. Kranakis, and N. Taleb. Patrolling trees with mobile robots. In *International Symposium on Foundations and Practice of Security*, pages 331–344. Springer, 2016.

[38] M.B. Dias. *Traderbots: A new paradigm for robust and efficient multirobot coordination in dynamic environments*. PhD thesis, Carnegie Mellon University Pittsburgh, 2004.

[39] M.B. Dias, R. Zlot, N. Kalra, and A. Stentz. Market-based multirobot coordination: A survey and analysis. *Proceedings of the IEEE*, 94(7):1257–1270, 2006.

[40] D. Dionne and C. A. Rabbath. Multi-UAV Decentralized Task Allocation with Intermittent Communications: the DTC algorithm. In *2007 American Control Conference*, pages 5406–5411, July 2007.

[41] L. E. Dubins. On curves of minimal length with a constraint on average curvature, and with prescribed initial and terminal positions and tangents. *American Journal of Mathematics*, 79(3):497–516, 1957.

[42] D. Eppstein. Arboricity and bipartite subgraph listing algorithms. *Information processing letters*, 51(4):207–211, 1994.

[43] E. Erel and S. C. Sarin. A survey of the assembly line balancing procedures. *Production Planning & Control*, 9(5):414–434, 1998.

[44] L. Evers, A.I. Barros, H. Monsuur, and A. Wagelmans. Online stochastic UAV mission planning with time windows and time-sensitive targets. *European Journal of Operational Research*, 238(1):348–362, 2014.

[45] A. Felicia and S. Sharif. A review on educational robotics as assistive tools for learning mathematics and science. *International Journal of Computer Science Trends and Technology*, 2(2):62–84, 2014.

[46] P. Flocchini, N. Santoro, G. Viglietta, and M. Yamashita. Rendezvous with constant memory. *Theoretical Computer Science*, 621:57–72, 2016.

[47] A. Franchi, L. Freda, G. Oriolo, and M. Vendittelli. The sensor-based random graph method for cooperative robot exploration. *Mechatronics, IEEE/ASME Transactions on*, 14(2):163–175, 2009.

[48] X. Fu, J. Pan, X. Gao, B. Li, J. Chen, and K. Zhang. Task allocation method for multi-uav teams with limited communication bandwidth. In *2018 15th International Conference on Control, Automation, Robotics and Vision (ICARCV)*, pages 1874–1878, November 2018.

[49] E. Galceran and M. Carreras. A survey on coverage path planning for robotics. *Robotics and Autonomous systems*, 61(12):1258–1276, 2013.

[50] B. P. Gerkey and M. J. Matarić. Sold!: auction methods for multirobot coordination. *Robotics and Automation, IEEE Transactions on*, 18(5):758–768, 2002.

[51] B. P. Gerkey and M. J. Matarić. Multi-robot task allocation: analyzing the complexity and optimality of key architectures. In *2003 IEEE International Conference on Robotics and Automation (Cat. No.03CH37422)*, volume 3, pages 3862–3868, September 2003.

[52] B. P. Gerkey and M. J. Matarić. A formal analysis and taxonomy of task allocation in multi-robot systems. *The International Journal of Robotics Research*, 23(9):939–954, 2004.

[53] S. Ghandi and E. Masehian. Review and taxonomies of assembly and disassembly path planning problems and approaches. *Computer-Aided Design*, 67-68:58–86, 2015.

[54] C. Ghedini, C. Secchi, C. H.C. Ribeiro, and L. Sabattini. Improving robustness in multi-robot networks. *IFAC-PapersOnLine*, 48(19):63 – 68, 2015.

[55] G. Giakkoupis, F. Mallmann-Trenn, and H. Saribekyan. How to spread a rumor: Call your neighbors or take a walk? In *Proceedings of the 2019 ACM Symposium on Principles of Distributed Computing*, PODC '19, pages 24–33, New York, NY, USA, 2019.

[56] S. Gil, S. Kumar, D. Katabi, and D. Rus. Adaptive communication in multi-robot systems using directionality of signal strength. *The International Journal of Robotics Research*, 34(7):946–968, 2015.

[57] A. R. Girard, A. S. Howell, and J. K. Hedrick. Border patrol and surveillance missions using multiple unmanned air vehicles. In *2004 43rd IEEE Conference on Decision and Control (CDC) (IEEE Cat. No.04CH37601)*, volume 1, pages 620–625, December 2004.

[58] C. Goerzen, Z. Kong, and B. Mettler. A Survey of Motion Planning Algorithms from the Perspective of Autonomous UAV Guidance. *Journal of Intelligent and Robotic Systems*, 57(1):65, 2009.

[59] A. K. Gupta, H. Sadawarti, and A. K. Verma. Performance analysis of manet routing protocols in different mobility models. *International Journal of Information Technology and Computer Science (IJITCS)*, 5(6):73–82, 2013.

[60] F. Hadlock. Finding a maximum cut of a planar graph in polynomial time. *SIAM Journal on Computing*, 4:221–225, 1975.

[61] L. A. Haidari, S. T. Brown, M. Ferguson, E. Bancroft, M. Spiker, A. Wilcox, R. Ambikapathi, V. Sampath, D. L. Connor, and B. Y. Lee. The economic and operational value of using drones to transport vaccines. *Vaccine*, 34(34):4062 – 4067, 2016.

[62] N. Hazon and G.A. Kaminka. On redundancy, efficiency, and robustness in coverage for multiple robots. *Robotics and Autonomous Systems*, 56(12):1102–1114, 2008.

[63] M. R. Henzinger, S. Rao, and H. N. Gabow. Computing vertex connectivity: New bounds from old techniques. *Journal of Algorithms*, 34(2):222 – 250, 2000.

[64] G. A. Hollinger and S. Singh. Multirobot coordination with periodic connectivity: Theory and experiments. *IEEE Transactions on Robotics*, 28(4):967–973, 2012.

[65] M. A. Hsieh, A. Cowley, V. Kumar, and C. J. Taylor. Maintaining network connectivity and performance in robot teams. *Journal of field robotics*, 25(1-2):111–131, 2008.

[66] R. I. Bor-Yaliniz, A. El-Keyi, and H. Yanikomeroglu. Efficient 3-D placement of an aerial base station in next generation cellular networks. In *2016 IEEE international conference on communications (ICC)*, pages 1–5. IEEE, 2016.

[67] P. Jiménez. Survey on assembly sequencing: a combinatorial and geometrical perspective. *Journal of Intelligent Manufacturing*, 24(2):235–250, 2013.

[68] A. E. Jimenez-Cano, J. Martin, G. Heredia, A. Ollero, and R. Cano. Control of an aerial robot with multi-link arm for assembly tasks. In *2013 IEEE International Conference on Robotics and Automation*, pages 4916–4921, May 2013.

[69] E. Kalantari, M. Z. Shakir, H. Yanikomeroglu, and A. Yongacoglu. Backhaul-aware robust 3D drone placement in 5G+ wireless networks. In *2017 IEEE international conference on communications workshops (ICC workshops)*, pages 109–114. IEEE, May 2017.

[70] E. Kalantari, H. Yanikomeroglu, and A. Yongacoglu. On the number and 3D placement of drone base stations in wireless cellular networks. In *2016 IEEE 84th Vehicular Technology Conference (VTC-Fall)*, pages 1–6. IEEE, September 2016.

[71] B. Kannan and L. E. Parker. Metrics for quantifying system performance in intelligent, fault-tolerant multi-robot teams. In *2007 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 951–958, October 2007.

[72] R. M. Karp. *Reducibility among Combinatorial Problems*, pages 85–103. Springer US, Boston, MA, 1972.

[73] L. Kavraki, J.C. Latombe, and R. H. Wilson. On the complexity of assembly partitioning. *Information Processing Letters*, 48(5):229 – 235, 1993.

[74] A. Khamis, A. Hussein, and A. Elmogy. *Multi-robot Task Allocation: A Review of the State-of-the-Art*, pages 31–51. Springer International Publishing, Cham, 2015.

[75] A. King. The future of agriculture. *Nature*, 544(7651):S21–S23, 2017.

[76] D. Kingston, R. W. Beard, and R. S. Holt. Decentralized perimeter surveillance using a team of uavs. *IEEE Transactions on Robotics*, 24(6):1394–1404, 2008.

[77] A. Kolling and A. Kleiner. Multi-UAV Motion Planning for Guaranteed Search. In *Proceedings of the 2013 International Conference on Autonomous Agents and Multi-agent Systems*, AAMAS '13, pages 79–86, Richland, SC, 2013.

[78] G. A. Korsah, A. Stentz, and M.B. Dias. A comprehensive taxonomy for multi-robot task allocation. *The International Journal of Robotics Research*, 32(12):1495–1512, 2013.

[79] H. A. Kurdi and J. P. How. Dynamic task allocation in an autonomous multi-UAV mission (US Patent 10,203,701), February 12 2019.

[80] J.P. Laumond, editor. *Robot Motion Planning and Control*. Springer-Verlag, Berlin, Heidelberg, 1998.

[81] T. Lemaire, R. Alami, and S. Lacroix. A distributed tasks allocation scheme in multi-UAV context. In *Robotics and Automation, 2004. Proceedings. ICRA '04. 2004 IEEE International Conference on*, volume 4, pages 3622–3627, April 2004.

[82] K. Lerman, C. Jones, A. Galstyan, and M. J. Matarić. Analysis of Dynamic Task Allocation in Multi-Robot Systems. *The International Journal of Robotics Research*, 25(3):225–241, 2006.

[83] D. A. Levin and Y. Peres. *Markov Chains and Mixing Times*. MBK. American Mathematical Society, 2017.

[84] S. Lin and B. W. Kernighan. An effective heuristic algorithm for the traveling-salesman problem. *Operations Research*, 21(2):498–516, 1973.

[85] L. Lovász. Random walks on graphs. *Combinatorics, Paul erdos is eighty*, 2(1-46):4, 1993.

[86] J. Lyu, Y. Zeng, R. Zhang, and T. J. Lim. Placement optimization of UAV-mounted mobile base stations. *IEEE Communications Letters*, 21(3):604–607, 2016.

[87] A. Machado, G. Ramalho, J.D. Zucker, and A. Drogoul. Multi-agent patrolling: An empirical analysis of alternative architectures. In *International workshop on multi-agent systems and agent-based simulation*, pages 155–170. Springer, 2002.

[88] P. T. Meijer. Connectivities and diameters of circulant graphs. Master's thesis, Simon Fraser University, 1991.

[89] C. Merengo, F. Nava, and A. Pozzetti. Balancing and sequencing manual mixed-model assembly lines. *International Journal of Production Research*, 37(12):2835–2860, 1999.

[90] 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, 2012.

[91] A. Merwaday and I. Guvenc. UAV assisted heterogeneous networks for public safety communications. In *2015 IEEE Wireless Communications and Networking Conference Workshops (WCNCW)*, pages 329–334, March 2015.

[92] U. R. Mogili and B. B.V.L. Deepak. Review on application of drone systems in precision agriculture. *Procedia Computer Science*, 133:502 – 509, 2018.

[93] M. Mozaffari, W. Saad, M. Bennis, and M. Debbah. Drone small cells in the clouds: Design, deployment and performance analysis. In *2015 IEEE Global Communications Conference (GLOBECOM)*, pages 1–6. IEEE, 2015.

[94] M. Mozaffari, W. Saad, M. Bennis, and M. Debbah. Efficient deployment of multiple unmanned aerial vehicles for optimal wireless coverage. *IEEE Communications Letters*, 20(8):1647–1650, 2016.

[95] M. Mozaffari, W. Saad, M. Bennis, and M. Debbah. Mobile Internet of Things: Can UAVs provide an energy-efficient mobile architecture? In *2016 IEEE global communications conference (GLOBECOM)*, pages 1–6. IEEE, 2016.

[96] M. Mozaffari, W. Saad, M. Bennis, and M. Debbah. Optimal transport theory for power-efficient deployment of unmanned aerial vehicles. In *2016 IEEE international conference on communications (ICC)*, pages 1–6. IEEE, 2016.

[97] M. Mozaffari, W. Saad, M. Bennis, and M. Debbah. Unmanned aerial vehicle with underlaid device-to-device communications: Performance and tradeoffs. *IEEE Transactions on Wireless Communications*, 15(6):3949–3963, 2016.

[98] M. Mozaffari, W. Saad, M. Bennis, and M. Debbah. Wireless communication using unmanned aerial vehicles (UAVs): Optimal transport theory for hover time optimization. *IEEE Transactions on Wireless Communications*, 16(12):8052–8066, 2017.

[99] R. M. Murray, Z. Li, and S. S. Sastry. *A mathematical introduction to robotic manipulation*. CRC Press, 2nd edition, 2018.

[100] M. Nickels and C. J. Cullen. Mathematical Thinking and Learning Through Robotics Play for Children With Critical Illness: The Case of Amelia. *Journal for Research in Mathematics Education*, 48(1):22–77, 2017.

[101] A. Otto, N. Agatz, J. Campbell, B. Golden, and E. Pesch. Optimization approaches for civil applications of unmanned aerial vehicles (UAVs) or aerial drones: A survey. *Networks*, 72(4):411–458, 2018.

[102] F. Pasqualetti, A. Franchi, and F. Bullo. On cooperative patrolling: Optimal trajectories, complexity analysis, and approximation algorithms. *Robotics, IEEE Transactions on*, 28(3):592–606, 2012.

[103] F. Potra. Mathematics and robotics (lecture). `https://www.maa.org/meetings/calendar-events/mathematics-and-robotics`, 2013.

[104] A. Puri. A Survey of Unmanned Aerial Vehicles (UAV) for Traffic Surveillance. Department of computer science and engineering, University of South Florida, 2005.

[105] S. Quinlan. Efficient distance computation between non-convex objects. In *Robotics and Automation, International Conference on*, pages 3324–3329. IEEE, 1994.

[106] B. Rabta, C. Wankmüller, and G. Reiner. A drone fleet model for last-mile distribution in disaster relief operations. *International Journal of Disaster Risk Reduction*, 28:107 – 112, 2018.

[107] M. F. F. Rashid, W. Hutabarat, and A. Tiwari. A review on assembly sequence planning and assembly line balancing optimisation using soft computing approaches. *The International Journal of Advanced Manufacturing Technology*, 59(1-4):335–349, 2012.

[108] P. Révész. *Random Walk in Random and Non-Random Environments*. World Scientific, 3rd edition, 2013.

[109] N. Roy and G. Dudek. Collaborative robot exploration and rendezvous: Algorithms, performance bounds and observations. *Autonomous Robots*, 11(2):117–136, 2001.

[110] F. Ruggiero, V. Lippiello, and A. Ollero. Aerial manipulation: A literature review. *IEEE Robotics and Automation Letters*, 3(3):1957–1964, 2018.

[111] J.R. Sack and J. Urrutia, editors. *Handbook of Computational Geometry*. North-Holland Publishing Co., Amsterdam, The Netherlands, The Netherlands, 2000.

[112] A. Sawalmeh, N. S. Othman, H. Shakhatreh, and A. Khreishah. Providing wireless coverage in massively crowded events using UAVs. In *2017 IEEE 13th Malaysia International Conference on Communications (MICC)*, pages 158–163. IEEE, 2017.

[113] A. Scholl and C. Becker. State-of-the-art exact and heuristic solution procedures for simple assembly line balancing. *European Journal of Operational Research*, 168(3):666 – 693, 2006.

[114] A. Sempere, D. Llorente, I. Maza, and A. Ollero. *Local Heuristics Analysis in the Automatic Computation of Assembly Sequences for Building Structures with Multiple Aerial Robots*, volume 252 of *Advances in Intelligent Systems and Computing*, pages 87–101. Springer International Publishing, 2014.

[115] E. Semsch, M. Jakob, D. Pavlicek, and M. Pechoucek. Autonomous UAV Surveillance in Complex Urban Environments. In *Proceedings of the 2009 IEEE/WIC/ACM International Joint Conference on Web Intelligence and Intelligent Agent Technology - Volume 02*, WI-IAT '09, pages 82–85, Washington, DC, USA, 2009.

[116] S. A. W. Shah, T. Khattab, M. Z. Shakir, and M. O. Hasna. A Distributed Approach for Networked Flying Platform Association with Small Cells in 5G+ Networks. In *GLOBECOM 2017 - 2017 IEEE Global Communications Conference*, pages 1–7, December 2017.

[117] H. Shakhatreh, A. Khreishah, A. Alsarhan, I. Khalil, A. Sawalmeh, and N. S. Othman. Efficient 3D placement of a UAV using particle swarm optimization. In *2017 8th International Conference on Information and Communication Systems (ICICS)*, pages 258–263. IEEE, 2017.

[118] H. Shakhatreh, A. Khreishah, J. Chakareski, H.B. Salameh, and I. Khalil. On the continuous coverage problem for a swarm of UAVs. In *2016 IEEE 37th Sarnoff Symposium*, pages 130–135. IEEE, 2016.

[119] H. Shakhatreh, A. Khreishah, and B. Ji. Providing wireless coverage to high-rise buildings using uavs. In *2017 IEEE International Conference on Communications (ICC)*, pages 1–6. IEEE, 2017.

[120] H. Shakhatreh, A. Khreishah, and I. Khalil. Indoor mobile coverage problem using UAVs. *IEEE Systems Journal*, 12(4):3837–3848, 2018.

[121] H. Shakhatreh, A. Khreishah, N. S. Othman, and A. Sawalmeh. Maximizing indoor wireless coverage using UAVs equipped with directional antennas. In *2017 IEEE 13th Malaysia International Conference on Communications (MICC)*, pages 175–180. IEEE, 2017.

[122] H. Shakhatreh, A. H. Sawalmeh, A. Al-Fuqaha, Z. Dou, E. Almaita, I. Khalil, N. S. Othman, A. Khreishah, and M. Guizani. Unmanned Aerial Vehicles (UAVs): A Survey on Civil Applications and Key Research Challenges. *IEEE Access*, 7:48572–48634, 2019.

[123] E. Shang. Introduction to markov chains and markov chain mixing. `http://math.uchicago.edu/~may/REU2018/REUPapers/Shang.pdf`, 2018.

[124] K. Shang, S. Karungaru, Z. Feng, L. Ke, and K. Terada. A GA-ACO hybrid algorithm for the multi-UAV mission planning problem. In *2014 14th International Symposium on Communications and Information Technologies (ISCIT)*, pages 243–248, September 2014.

[125] W. Sheng, Q. Yang, J. Tan, and N. Xi. Distributed multi-robot coordination in area exploration. *Robotics and Autonomous Systems*, 54(12):945–955, 2006.

[126] T. Shima, S. J. Rasmussen, A. G. Sparks, and K. M. Passino. Multiple task assignments for cooperating uninhabited aerial vehicles using genetic algorithms. *Computers and Operations Research*, 33(11):3252 – 3269, 2006.

[127] E. M. Silk and C. D. Schunn. Using robotics to teach mathematics: Analysis of a curriculum designed and implemented. In *American Society for Engineering Education Annual Conference, Pittsburgh, PA*, 2008.

[128] V. Srivastava, F. Pasqualetti, and F. Bullo. Stochastic surveillance strategies for spatial quickest detection. *The International Journal of Robotics Research*, 32(12):1438–1458, 2013.

[129] D. Stehlé and P. Zimmermann. A binary recursive gcd algorithm. In *Algorithmic number theory*, volume 3076 of *Lecture Notes in Comput. Sci.*, pages 411–425. Springer, Berlin, 2004.

[130] S. Tabachnikov. *Geometry and billiards*, volume 30. American Mathematical Society, Providence, RI, 2005.

[131] L. Tang, J. Liu, A. Rong, and Z. Yang. A multiple traveling salesman problem model for hot rolling scheduling in Shanghai Baoshan Iron & Steel Complex. *European Journal of Operational Research*, 124(2):267 – 282, 2000.

[132] D. Tardioli, A .R. Mosteo, L. Riazuelo, J. L. Villarroel, and L. Montano. Enforcing network connectivity in robot team missions. *The International Journal of Robotics Research*, 29(4):460–480, 2010.

[133] W. T. L. Teacy, J. Nie, S. McClean, and G. Parr. Maintaining connectivity in UAV swarm sensing. In *2010 IEEE Globecom Workshops*, pages 1771–1776, December 2010.

[134] M. Tognon, H.A. Tello Chávez, E. Gasparin, Q. Sablé, D. Bicego, A. Mallet, M. Lany, G. Santi, B. Revaz, J. Cortés, and A. Franchi. A Truly Redundant Aerial Manipulator System with Application to Push-and-Slide Inspection in Industrial Plants. *IEEE Robotics and Automation Letters*, 4(2):1846 – 1851, 2019.

[135] P. Tripicchio, M. Satler, G. Dabisias, E. Ruffaldi, and C. A. Avizzano. Towards smart farming and sustainable agriculture with drones. In *2015 International Conference on Intelligent Environments*, pages 140–143, July 2015.

[136] M.Á. Trujillo, J.R. Martínez-de Dios, C. Martín, A. Viguria, and A. Ollero. Novel aerial manipulator for accurate and robust industrial ndt contact inspection: A new tool for the oil and gas inspection industry. *Sensors*, 19(6):1305, 2019.

[137] A. Tsalatsanis, A. Yalcin, and K. P. Valavanis. Dynamic task allocation in cooperative robot teams. *Robotica*, 30:721–730, 2012.

[138] G. Tuna, B. Nefzi, and G. Conte. Unmanned aerial vehicle-aided communications system for disaster recovery. *Journal of Network and Computer Applications*, 41:27–36, 2014.

[139] K. P. Valavanis and G. J Vachtsevanos, editors. *Handbook of Unmanned Aerial Vehicles*. Springer, 2015.

[140] J. Valente, D. Sanz, A. Barrientos, J. del Cerro, Á. Ribeiro, and C. Rossi. An air-ground wireless sensor network for crop monitoring. *Sensors*, 11(6):6088–6108, 2011.

[141] C. Wang and E. P. Chan. Finding the minimum visible vertex distance between two non-intersecting simple polygons. In *Computational Geometry, 2nd Annual Symposium on*, pages 34–42. ACM, 1986.

[142] C. Wang, F. Ma, J. Yan, D. De, and S.K. Das. Efficient aerial data collection with UAV in large-scale wireless sensor networks. *International Journal of Distributed Sensor Networks*, 11(11):286080, 2015.

[143] X. Wang, S. Poikonen, and B. Golden. The vehicle routing problem with drones: several worst-case results. *Optimization Letters*, 11(4):679–697, 2017.

[144] X. Wang, Y. Zhang, L. Wang, D. Lu, and G. Zeng. Robustness evaluation method for unmanned aerial vehicle swarms based on complex network theory. *Chinese Journal of Aeronautics*, 2019.

[145] A. Whitbrook, Q. Meng, and P. W. H. Chung. Addressing robustness in time-critical, distributed, task allocation algorithms. *Applied Intelligence*, 49(1):1–15, 2019.

[146] A. F. T. Winfield. Distributed sensing and data collection via broken ad hoc wireless connected networks of mobile robots. In *Distributed Autonomous Robotic Systems 4*, pages 273–282. Springer, 2000.

[147] A. Xu, C. Viriyasuthee, and I. Rekleitis. Optimal complete terrain coverage using an unmanned aerial vehicle. In *Robotics and Automation (ICRA), 2011 IEEE International Conference on*, pages 2513–2519, May 2011.

[148] D. Yang, Q. Wu, Y. Zeng, and R. Zhang. Energy tradeoff in ground-to-UAV communication via trajectory design. *IEEE Transactions on Vehicular Technology*, 67(7):6721–6726, 2018.

[149] L. Yang, J. Qi, D. Song, J. Xiao, J. Han, and Y. Xia. Survey of Robot 3D Path Planning Algorithms. *Journal of Control Science and Engineering*, 2016:5–27, 2016.

[150] E. Yanmaz, C. Costanzo, C. Bettstetter, and W. Elmenreich. A discrete stochastic process for coverage analysis of autonomous UAV networks. In *2010 IEEE Globecom Workshops*, pages 1777–1782. IEEE, 2010.

[151] S. Yoon and C. Qiao. A New Search Algorithm using Autonomous and Cooperative Multiple Sensor Nodes. In *IEEE INFOCOM 2007 - 26th IEEE International Conference on Computer Communications*, pages 937–945, May 2007.

[152] C. Yuan, Y. Zhang, and Z. Liu. A survey on technologies for automatic forest fire monitoring, detection, and fighting using unmanned aerial vehicles and remote sensing techniques. *Canadian Journal of Forest Research*, 45(7):783–792, 2015.

[153] C. Zhan, Y. Zeng, and R. Zhang. Energy-efficient data collection in UAV enabled wireless sensor network. *IEEE Wireless Communications Letters*, 7(3):328–331, 2017.

[154] Y. Zhou, N. Cheng, N. Lu, and X. S. Shen. Multi-UAV-Aided Networks: Aerial-Ground Cooperative Vehicular Networking Architecture. *IEEE Vehicular Technology Magazine*, 10(4):36–44, 2015.

[155] M. Zhu, Z. Cai, D. Zhao, J. Wang, and M. Xu. Using multiple unmanned aerial vehicles to maintain connectivity of manets. In *2014 23rd International Conference on Computer Communication and Networks (ICCCN)*, pages 1–7. IEEE, 2014.