Cooperative navigation of a fleet of mobile robots

Daravuth Koung

CENTRALE NANTES

THÈSE DE DOCTORAT DE

L'ÉCOLE CENTRALE DE NANTES

ÉCOLE DOCTORALE N° 601
Mathématiques et Sciences et Technologies de l'Information et de la Communication
Spécialité : *Automatique, productique et robotique*

Par
Daravuth KOUNG

Cooperative navigation of a fleet of mobile robots

Thèse présentée et soutenue à l'Ecole Centrale de Nantes, le 19 octobre 2022
Unité de recherche : UMR 6004, Laboratoire des Sciences du Numérique de Nantes (LS2N)

Rapporteurs avant soutenance :

Charles LESIRE — Chercheur HDR, ONERA Toulouse
Roland LENAIN — Directeur de recherche, INRAE, Aubière

Composition du Jury :

Président :	Philippe MARTINET	Directeur de Recherche, Centre INRIA d'Université Côte d'Azur
Examinateurs :	Olivier SIMONIN	Professeur des universités, INSA Lyon
	Vincent FREMONT	Professeur des universités, Ecole Centrale de Nantes
Dir. de thèse :	Isabelle FANTONI	Directrice de recherche CNRS, Ecole Centrale de Nantes
Co-encadrant :	Olivier KERMORGANT	Maître de conférences, Ecole Centrale de Nantes
Co-encadrante :	Lamia BELOUAER	Responsable développement logiciel, E-COBOT, Carquefou

Invité :

Sébastien ECAULT — Président, E-COBOT, Carquefou

Acknowledgement

I am so thankful to those who have helped and contributed to the achievement of this PhD thesis. First of all, I would like to express my deepest gratitude to my supervisors: Isabelle Fantoni, Olivier Kermorgant, and Lamia Belouaer for their continuous support, motivation, and immense guidance. I could not ask for better mentors for my PhD study.

My sincere thanks also go to Sébastien Ecault, CEO of E-COBOT who has initiated this PhD collaboration project. And Sébastien Briot, head of the ARMEN team who has given me access to the laboratory and research facilities.

I thank my fellow labmates for the countless discussions and encouragements. I also thank my colleagues at E-COBOT who have supported me during this PhD journey.

Finally, yet importantly, I am glad to denote my appreciation to my parents and other supportive people including lecturers, friends, and interns whose names were not mentioned above, but have held significant impacts on the successfulness of this thesis.

Daravuth Koung

Table of Contents

List of Figures

List of Tables

INTRODUCTION

With recent rapid advancements in technology, the fourth industrial revolution, industrial 4.0, is born with the emphasis on smart factory [1]. With this trend, warehouses and distribution centers (DC) start to adopt automation and robotics in their daily operations. In fact, they can leverage multi-robot system (MRS) technology to greatly benefit from its robustness and efficiency compared to single-robot systems [GM12]. As the name suggested, MRS is a robotics system formed by multiple robots; they may or may not cooperate and communicate with each other to accomplish certain tasks. The tasks can be found in a wide range of applications. Patrolling [Pop+17] is one of them, which requires robots to visit certain places over and over again. The task is to divide roles and coverage areas between multiple robots. Exploration and mapping is also a subject of interest applying to MRS. The benefits of having multiple robots performing this task are shown in [Gif+10]. On the other hand, a task that corresponds the most to the automatizing warehouses and DC is the load transportation task. Indeed, not only it can improve efficiency, but also workers' safety. According to the data from the U.S. Bureau of Labor Statistics (BLS), the rate of recordable illness and injury cases in the warehousing and storage sector was 4.8 out of 100 workers in 2020 [2].

Formation control is a basis of the MRS when dealing with transportation tasks, especially for a heavy load that requires more than one robot working together to move it. Formation control can be seen as a problem of driving robots to a desired geometric shape, which is, normally, defined by inter-distances between robots in the group [Mia+18]. Depending on the hardware setups and transporting scenarios, the team's formation can be a rigid or flexible shape. For instance, the formation in [WS16] does not need to be rigid because of the gripper attached to each robot. In contrast, our goal is to transport the load by placing it directly on top of the robots. The challenge of our aim is to be able to use any commercial off-the-shelf robot for the task without needing to heavily modify or customize the robots. Therefore, a rigid formation is required for such a mission.

Formation control enables a group of robots to cooperatively carry a load. However,

1. https://www.i-scoop.eu/industry-4-0/
2. https://www.bls.gov/iag/tgs/iag493.htm

it does not decide which robot should be chosen for the formation team; this is where multi-robot task allocation (MRTA) is needed. MRTA problems are NP-hard optimization problems whose aim is to assign robots to tasks in an optimal manner [Hus+18]. The problem is usually expressed as a cost function that is defined based on the context and requirements of the system. Since our main application of interest is logistics in warehouses, tasks to be allocated can require either single (simple pick-and-place mission) or multiple robots (heavy/huge load transportation). Tasks may be defined as a static list to be allocated at once as well as dynamically added online during the operation. Total execution time, which includes time to do allocation and time to complete the mission, is considered the optimizing parameter. Two different allocation approaches are studied in order to determine the most suitable for these constraints: Contract Net Protocol (CNP) and Tabu Search (TS). CNP is an auction-based approach that consists of an auctioneer, whose job is to send out proposals of tasks, and a group of robots, which send their bids back to the auctioneer for the tasks [NMS15]. TS is a local search method that allows the search to step on worse solution neighbors, thus escaping the local minimum. A tabu list is introduced to store sets of banned solutions for several iterations to avoid cycling [LRV14].

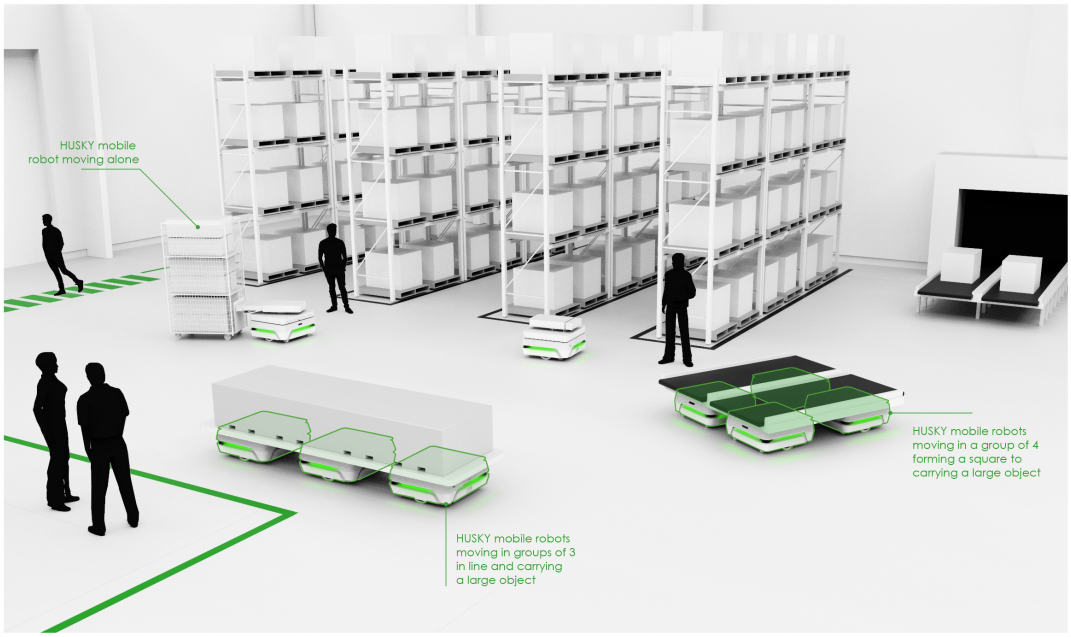


Figure 1 – A use case scenario of the MRS.

A practical use case concerning the application in the warehouse is shown in Figure 1. In a given environment, numerous robots co-exist in order to help the operators moving things from one point to another. This is where task allocation is needed to adequately match a robot with a mission. Moreover, if the load to be placed on top is heavy or huge, thus a group of robots is needed, the formation control is required in order to avoid the load to slip off. The formation can be in various shapes and sizes of the number of robots. In addition to cooperative navigation, the team also has to be able to reactively avoid any obstacles that are not already mapped in the environment.

This thesis tries to answer this use case. The goal of this PhD research is to develop and implement in an industrial environment a cooperative load-transporting system by leveraging the multi-robot system. A group of nonholonomic mobile robots is used to carry the load on top. Since there are usually more robots than required for a task in a real-world scenario, multi-robot task allocation is required to efficiently choose adequate robots. The contributions of this thesis are as follows: 1) adaptation of a consensus flocking algorithm to formation control for the nonholonomic system, 2) improvement to the obstacle avoidance of the former algorithm such that the formation remains rigid while evading the obstacle, 3) proposition of an optimization-based formation control that decomposes the transportation task into multiple cost objectives with different priority levels. Hierarchy quadratic programming is then used to solve for an optimal formation control that can preserve the shape during navigation, and 4) comparison of two task allocation methods: CNP and TS, in order to determine a suitable approach for our application.

This thesis acts as a testament to the effectiveness of the collaboration between academic lab and company. In fact, this research project is proposed to the LS2N lab [3] by E-COBOT [4], a manufacturer of mobile robots whose mission is to provide flexible and reconfigurable logistics solutions for warehouses and factories. Parts of this work will be implemented directly to the robot fleet of E-COBOT, named Husky. It is important to show that the research done in the scientific lab can be practically deployed in a commercially viable way.

The thesis starts with an overview of the state-of-the-art of multi-robot system. In this first chapter, we present different types of control architectures as well as control strategies for both the formation and task allocation. Chapter 2 presents the work done for

3. https://www.ls2n.fr
4. https://e-cobot.com

the formation control. This chapter includes the two proposed controllers for formation: consensus-based and optimization-based. Simulation and experimental results of these two approaches are also presented in this chapter. The comparison work of task allocation can be found in chapter 3, where details of the two approaches are introduced along with the simulations of different study cases. Last but not least, the overall conclusion and perspectives for future work are discussed in the final chapter.

Publications

Journal article

Daravuth Koung, Olivier Kermorgant, Isabelle Fantoni, and Lamia Belouaer, "Cooperative Multi-Robot Object Transportation System Based on Hierarchical Quadratic Programming", *IEEE Robotics and Automation Letters*, Oct., 2021, vol. 6, pp. 6466-6472. (This paper was also selected for presentation at the 2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS))

Conference proceeding

Daravuth Koung, Isabelle Fantoni, Olivier Kermorgant, and Lamia Belouaer, "Consensus-based formation control and obstacle avoidance for nonholonomic multi-robot system", *International Conf. on Control, Automation, Robotics and Vision (ICARCV)*, Dec., 2020.

Poster session

Daravuth Koung, Isabelle Fantoni, Olivier Kermorgant, and Lamia Belouaer, "Cooperative navigation of mobile robots fleet", *Journées Francophones sur la Planification, la Décision et l'Apprentissage pour la conduite de systèmes (JFPDA)*, July, 2019.

Chapter 1

State of the art

Contents

1.1 Introduction

In this chapter, an overview of the work related to the multi-robot system (MRS) is presented. We start with a classification of control architecture used in the MRS. Then, we present state-of-the-art techniques for formation control, which is followed by an introduction to various load handling strategies. These strategies focus particularly on logistics problems, which corresponds to our application scenarios. Finally, a general review of the multi-robot task allocation methods is presented.

As the MRS is a broad field of research, the reviews of literature in the following sections focus on pure interaction between mobile robots. It does not discuss other forms of MRS such as collaborative robotic manipulators or human-robot interaction.

1.2 Control architecture

The research activities of the multi-robot system have been taken off as early as the late 80s; work done in [AMI89] and [Cal+90] are some of the earliest scientific researches in the field. Since then, many studies have been conducted; these works mainly fall into three control architectures [YJC13][VR21]: centralized, distributed, and hierarchical.

In this section, we discuss the differences between these control architectures including their advantages and disadvantages.

1.2.1 Centralized architecture

This architecture is based on a central control agent as shown in Figure 1.1. This controller can be an external server or a robot in the team. In order to coordinate and generate control command of the overall system, it requires the states of all robots in the group.

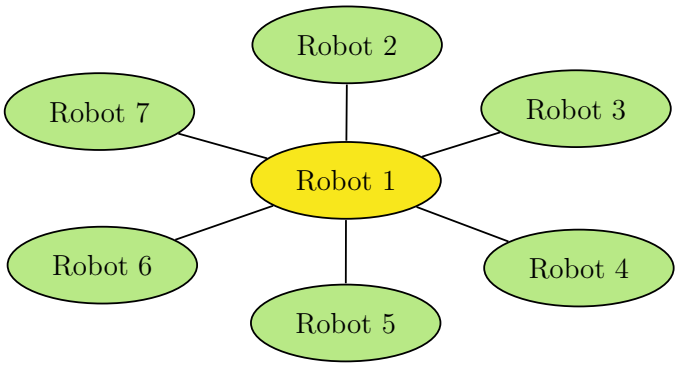


Figure 1.1 – A centralized control architecture, where the orange node is the control agent.

Since the controller receives every robot's state, the main advantage of this control structure is the global knowledge of each robot in the group, which is crucial for optimal decisions and performance. However, this architecture suffers an issue of robustness and reliability since the group of robots depends on a central controller. In addition, it, generally, requires higher computation power compared to other control architectures, which diminishes the system's scalability [YJC13].

1.2.2 Distributed architecture

This control architecture allows each robot to make its own decisions. As shown in Figure 1.2, this architecture is based on the idea of sharing information between neighbors. Each robot makes decisions independently with respect to others' states; there is no single-point controller. The exchanging information can be anything from velocity, and position to sensor data such as lidar scan points or camera images depending on the needs of the system.

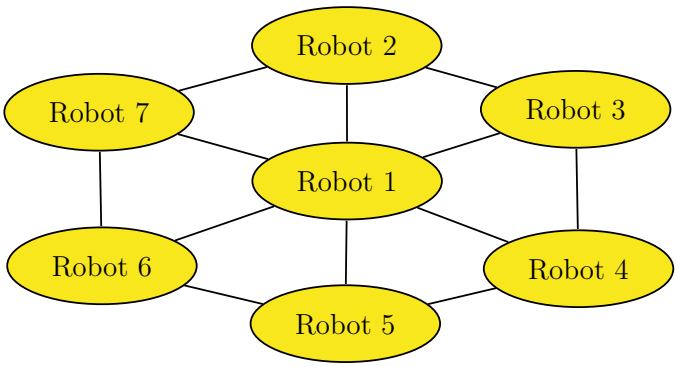


Figure 1.2 – A distributed control architecture.

Due to the nature of having no central controller, reliability is one of its main advantages. In fact, it responds better to changing or unknown environments. Moreover, it is also suitable for large and scalable systems because the communication is limited to within the interaction range between neighbors. However, this could lead to a solution that is often suboptimal in terms of stability and convergence rate [YJC13].

There is a growing interest in this type of control architecture; in some literature, however, this controller is referred to as *decentralized*. This could be because of the researchers' backgrounds and specialty differences.

1.2.3 Hierarchical architecture

This architecture is a mixture of the centralized and distributed control structure; there could be more than one central controller, where each controller is in charge of a local cluster of robots. Figure 1.3 illustrates a hierarchical architecture; there are two local central agents (*Robot 1* and *Robot 2*) for coordinating their respective cluster.

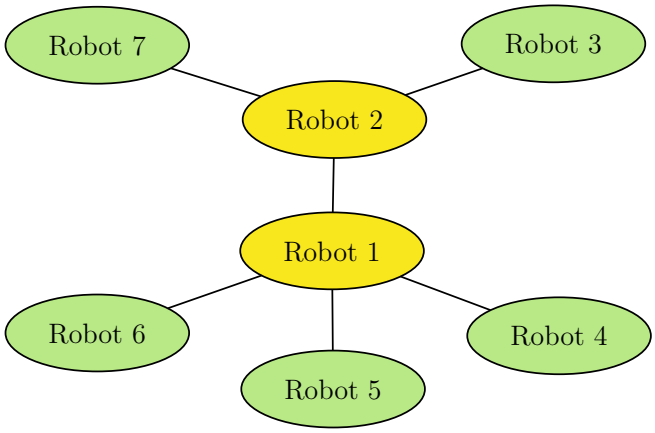


Figure 1.3 – A hierarchical control architecture.

This control approach is, generally, implemented in an environment where there exist numerous independent tasks; each task required an individual or a group of robots. Compared to distributed, this architecture is less robust; but requires less communication between robots. Similar to the distributed situation, this type of control is called, interchangeably, as *hybrid* in the literature.

1.3 Formation control

The formation is one of many applications of the MRS. Formation control can be seen as a problem of driving robots to a desired geometric shape, which is, normally, defined by the inter-distance between robots in the network [Mia+18]. Numerous control strategies can be, interestingly, seen in the literature. Some of which are [Hou+17]: leader-follower, virtual structure, and behavior-based approach, which we will explore in this section.

1.3.1 Leader-follower

Leader-follower approach is widely used for formation control due to its simplicity for implementation. In the control strategy, one or more robots are assigned to be the leader(s) of the group, which leaves the remaining members as followers. Figure 1.4 shows a simple one leader and one follower situation. The reference of the desired trajectory is given to the leader(s), and the local control law of each follower can be defined based on the desired relative position of that follower to its leader.

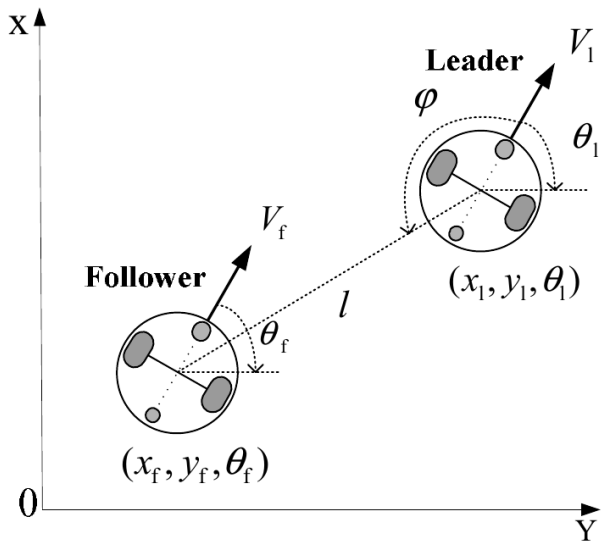


Figure 1.4 – A leader-follower example [LLL15].

Some work based on the basic configuration of a leader and a follower can be found in [LLL15], [Con+06], and [Wan+18]. In [Con+06], the formation is formed by defining a geometric framework based on the distance and angle between each follower and the leader. Instead of following the leader, the authors of [LLL15] introduce a virtual-following robot whose position is derived from the leader; then the problem is converted from formation to tracking control for the follower robots. In [Wan+18], the authors use a similar technique as [Con+06]; there is an obstacle avoidance in addition, and a bounded barrier function to ensure that the leader stays inside the follower's field of view. In [Wan+17a], a vision-based leader-follower approach is studied. The controller is an image-based visual servoing; control variables are the 2D coordinate of the leader detected by a pin-hole camera. The idea is to avoid any need for communication from the leader.

Stability analysis of such an approach is relatively straightforward as the controller of each follower is mostly defined based on the distance and bearing to the leader. On the other hand, this control strategy depends solely on the leader. It must not be failed or it can affect the whole formation. In addition, error propagation is common in this system. There is, generally, no communication feedback from each follower to the leader. As a result, the leader may leave its followers behind, for instance, when the leader moves too fast. The lack of ability to perform self-organization is also an issue in this approach.

1.3.2 Virtual structure

For the strategy of the virtual structure, a central controller presumes the entire group of robots as a single entity. The desired motion is defined for the whole formation, which is, then, transformed into the desired motion of each robot for individual tracking. Examples of such work can be found in [BWN09] and [CB15]; nonholonomic robot formation is performed using a virtual center, which is defined from a predefined reference trajectory. As illustrated in Figure 1.5a, the virtual center is used to determine the desired position for each robot at any time instance; then the robots track this position in Cartesian space following a control command from a centralized computer. In [LN11], this control approach is applied to fixed-wing UAVs. However, the virtual structure introduced is flexible in order to allow the formation to turn smoothly following the desired trajectory. Moreover, the virtual reference point does not have to always be at the center of the formation. As demonstrated in [YN08], *virtual vehicle* (VV) is introduced for each robot instead. With different desired positional relations between the robot and VV, the formation can be achieved by driving all the VV to converge to a common position as shown in Figure 1.5b.

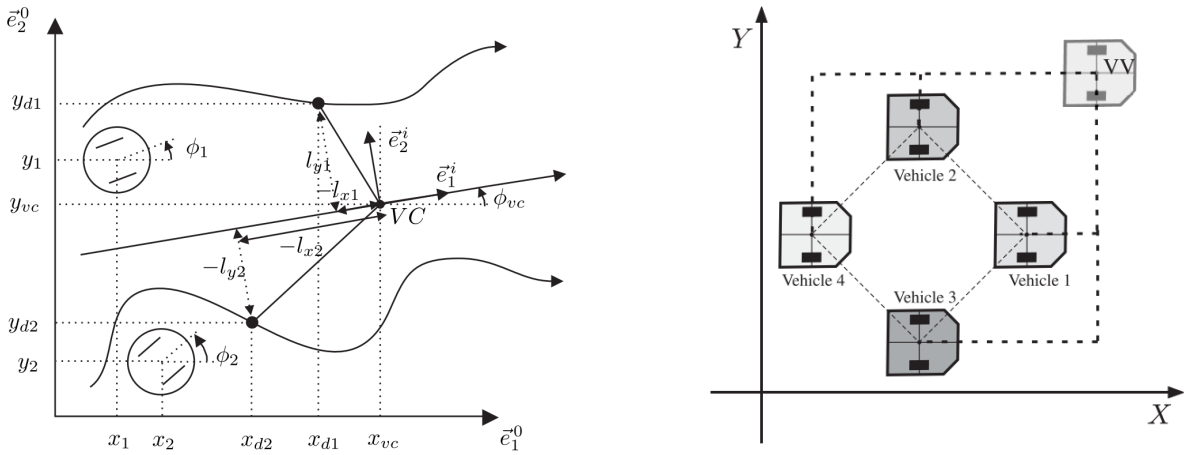


Figure 1.5 – A virtual structure using (a) virtual center approach [BWN09] and (b) virtual vehicle approach [YN08].

Since each robot has its own trajectory to follow, robots' behaviors can be simpler to describe. In addition, inter-robot communication is, generally, not required in such an approach. This is one of its benefits, but also a drawback because inter-robots collisions could occur in the presence of perturbations. Moreover, it is not a distributed approach,

thus suffering the previously mentioned disadvantages of a central controller.

1.3.3 Behavior-based

In the behavior-based approach, various desired behaviors corresponding to different situations are assigned to each robot. This strategy was first introduced by [Bro86], and we can still find this approach in recent work. Figure 1.6 illustrates the concept of this approach. The behaviors can be collision avoidance, obstacle avoidance, formation, go-to-goal, etc. These behaviors are then fused together based on their priority and importance.

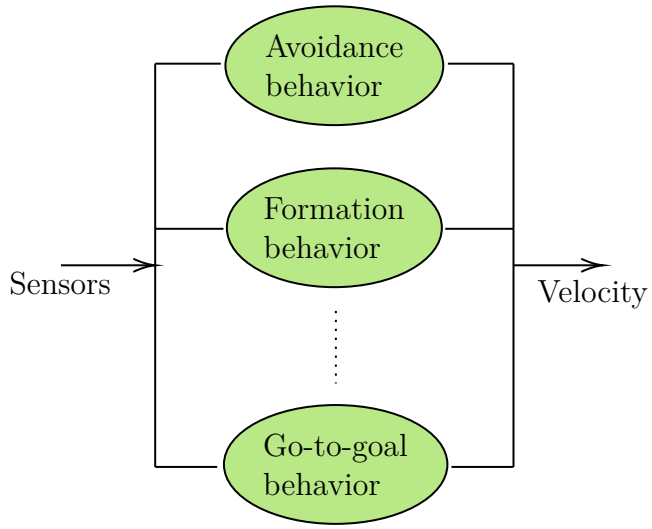


Figure 1.6 – A behavior-based controller with different behaviors.

In [Mar+13], the authors proposed a behavior-based framework of null-space-behavioral (NSB) control for patrolling application. This NSB approach is also implemented for formation control, which can be found in the work of [AFH14]. The NSB is used to solve a conflict of control velocity when a robot is a part of different groups that have the same or different formation tasks. On the other hand, a simple computation-free behavior-based control is studied in [SPB18]. In order to form a circle, each robot has four behavior states: 1) scouting: to find a leading robot; 2) chaining: to follow a moving robot; 3) hooking and looping: to close a loop; 4) merging: to integrate any lone robot to an existing circle.

The author of [Rey87] introduced three rules for flocking behavior, which are collision avoidance, velocity matching, and flock centering. The idea was to simulate the natural behavior of birds during the flight in the group, so-called flocking. Developed from these rules, recent flocking literature can be seen in [Olf06] and [SFZ19]. In order to keep a

coherent velocity, each robot's controller has a velocity control term that is based on consensus.

The consensus algorithm is a control strategy whose objective is to get to an agreement on some control features, for instance, the states of robots in the team [Hou+17]. [Wan+17b] is one of the recent literature that implements consensus for formation control. The authors introduce a weight function to the coefficient of consensus controller for nonholonomic wheeled robots. The formation is achieved once this weight converges to zero. In [Fu+19], the consensus is applied to second-order systems. The work also includes velocity constraints in form of a tangent and local velocity damping function.

The main advantages of this control strategy are the self-organization and scalability of the system. It does not require any central controller sinch each robot has its own set of rules and goals to follow. On the other hand, this approach suffers an issue of being complicated in terms of stability analysis, and a slower convergence rate compared to other methods.

1.3.4 Other approaches

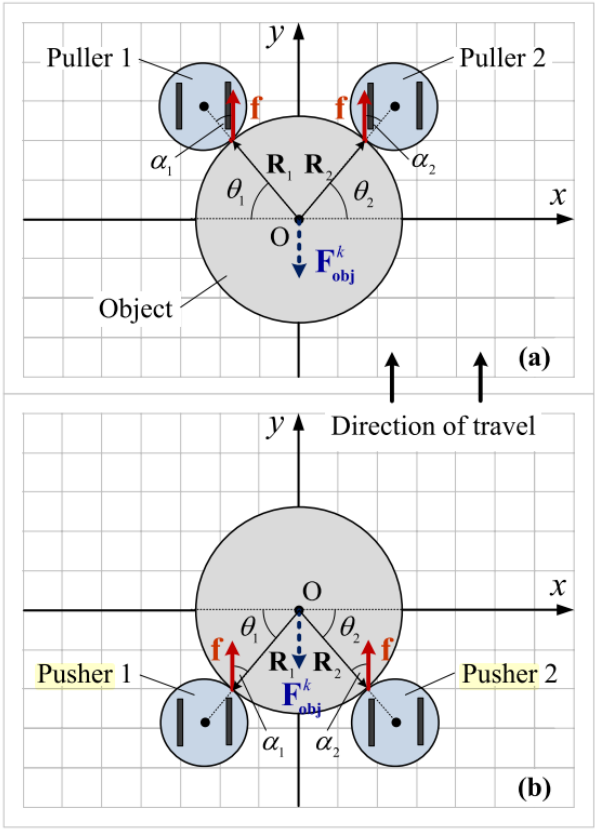
Apart from the presented well-known control strategies above, there are some other effective approaches such as optimization and machine learning.

Optimization is another method of interest for formation control. In [Liu+17], the authors propose a distributed model predictive control approach in which each mobile robot is considered as a subsystem. The neighboring subsystems are, then, coupled in the cost function where nash-optimization is used to solve for an optimal solution. The cost function includes path-following and formation. On the other hand, a centralized optimization-based control is proposed by [PAM20]. The authors define different tasks as equality and inequality constraints. The tasks consist of formation, obstacle avoidance, and trajectory tracking; they all have different levels of priority. The hierarchical quadratic programming framework is used in order to solve for an optimal solution that respects imposed constraints.

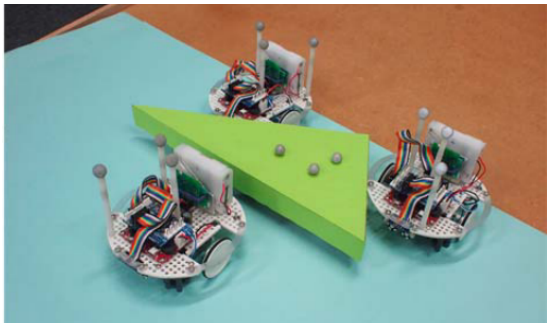
Last but not least, the use of machine learning techniques can also be found in the field of MRS. [LHW14] and [Wan+19] are some of the recent works in literature; the authors applied the reinforcement learning approach for formation control. In [LHW14], a set of formation behaviors is pre-defined. Each action contributes with a certain weight to the overall control; the reinforcement learning constantly adjusts these weight coefficients for optimal formation behavior. On the other hand, the authors of [Wan+19] propose a deep

neural networks framework that tries to generate a general formation pattern; it is based on trial-and-error feedback from each formation round in order to improve the formation strategy.

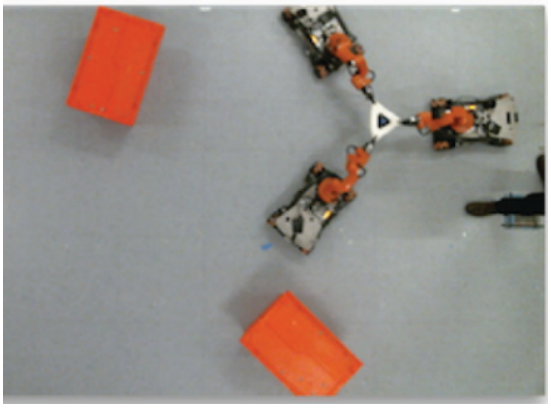
1.4 Load handling strategy



(a) pushing [Eoh+11]



(b) caging [Wan+20]



(c) grasping [ABR17]

Figure 1.7 – Difference types of load handling techniques.

As logistics is one of the emerging applications of the MRS, cooperative transportation task is a mission that has demands in real-world usage. To collectively move an object, a team of mobile robots can utilize one of the numerous load handling strategies; some of which include [TAA18]: pushing-only, caging, and grasping. Figure 1.7 shows the different types of load handling scenarios. For the pushing-only and caging strategy, there is no physical joint between the robots and the object. These two strategies are one of the easiest to implement. However, they can be inefficient due to frictional forces. The

terrain on which the robots operate has to be smooth. The same requirement also applies to the object's bottom surface. As the name suggested, all that is needed to be done to move an object is to straightforwardly push or, sometimes, pull that object (Figure 1.7a). This technique can be found in the work of [Eoh+11], [Che+15] and [ANT17]. When the object is surrounded by both pushing and pulling robots, as shown in Figure 1.7b, it is called caging. This approach is implemented in [PAM20], [Dai+16], and [Wan+20]. On the other hand, the grasping approach can be found in the work of [Yan+04], [WS16], and [ABR17]. In this case, the load is carried by manipulators that are on top of the mobile platforms; the manipulators may be either passive or active joints. Even though the rigidity of the formation's shape is not important, it requires additional and complex mechanisms added to the robots.

1.5 Task allocation

Multi-robot task allocation (MRTA) problem is solved differently depending on the type of tasks, robots, and assignation nature. Generally, there are three classifications of MRTA [GM04]. Table 1.1 shows a summary of this taxonomy. Single-robot (SR) task means only one robot is required to complete this task, while the Multi-robot (MR) may require multiple robots. Single-task (ST) robot represents a kind of robot that can execute only one task at a time, while the Multi-task (MT) counterpart can perform multiple tasks simultaneously. Instantaneous assignment (IA) assumes that tasks are independent of each other. With the available information on robots, tasks, and the environment, only instantaneous allocation is possible; there is no future assignment planned. Time-extended assignment (TA) allows a schedule allocation into the future. It requires more information, for instance, the dependencies between tasks.

Some of the practical problems that can be solved using MRTA techniques are the Multiple Traveling Salesman Problem (mTSP) and the Vehicle Routing Problem (VRP) [KSD13]. mTSP is a modified version of the classic Traveling Salesman problem (TSP). In TSP, the salesman has a list of towns to visit; the aim is to find the optimal path that would require the least traveling distance or time to complete the tour. All the towns have to be visited only once. Increased to m salesmen in mTSP, the goal remains the same; starting from the same point, each salesman has to visit at least one town before returning back to the start point. Applying to robotics, the town represents a destination point in the environment while each salesman is represented by a robot. The travel cost to be mini-

Table 1.1 – Taxonomy of MRTA.

Category	Type 1	Type 2
Tasks	SR: can be completed by one robot.	MR: requires multiple robots.
Robots	ST: can perform one task at a time.	MT: can perform multiple tasks at the same time.
Assignment	IA: a robot is assigned to one task at a time.	TA: a robot are given a schedule of tasks.

mized is distance and/or time. VRP is a more generalized version of mTSP. In VRP, the starting point of each vehicle does not have to be the same. Instead of having a list of towns or points to visit, it has sets of pickup and delivery location pairs. It aims to solve the problem of distributing goods or transporting passengers. Similar mTSP, vehicles will return to their own starting depot at the end.

Since the MRTA evolves around minimizing an objective function, different techniques for representing the cost function have been introduced [G L+05]:

— *MinSum*: to minimize the total cost of all robots.

— *MinMax*: to minimize the worst cost of a robot.

— *MinAve*: to minimize the average cost of all targets.

Numerous studies have been proposed in the literature to address the MRTA problems. We will present some of the recent work that mainly falls into two of the most widely used MRTA approaches [KHE15]: market-based or auction-based and optimization-based.

1.5.1 Auction-based

Inspired by economical studies, market-based relies on a mechanism called auction. Figure 1.8 demonstrates a process of the auction for market-based task allocation. A central server or one of the robots in the team can be an auctioneer in order to broadcast tasks to be done. Then, each individual robot submits its bids based on various factors such as distance to travel, time to execute, battery level, robot capability, etc. After all bids are received, the auctioneer can decide and pick the appropriate winner for the task.

The auction process can be divided into two categories [See+20]: single-item auction and combinatorial auction. These two are developed further to various variants, but the core definition remains. Single-item auctioning method allocates only a task at a time.

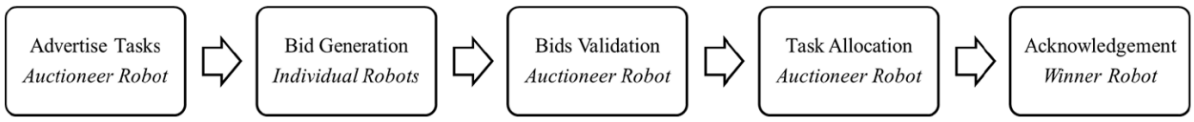


Figure 1.8 – A process of auction-based task allocation [See+20].

It is a simple and efficiently quick approach. However, optimality can not be guaranteed because some tasks may be close to each other and should be allocated to the same robot. This issue can be solved using the combinatorial auction. Since a set of positive synergy tasks are put to auction at once, the solution can be more optimal for the combinatorial than the single-item auction approach. However, the computational complexity of the system increases in this case as the auctioneer has to understand the nature of tasks and bundle them together.

A recent use of the auction-based approach can be found in [BSM20]. The authors rely solely on the distance between the robot's current position and the assigning task to compute the bid. In [BCD19], heuristics decisions are added to the auction process. Those heuristics include the closest robot to the starting of the mission; and the priority zone association of the robot. In [LCS15], the auction's objective is to maximize total payoffs, meaning maximum delivery of parts for assembly. Since this work is intended for a manufacturing environment, a battery constraint is also imposed in form of the total duration of tasks; this can not exceed the robot's status. In [SGC19], a single-item auctioning method is proposed for the heterogeneous system. The authors focus on minimizing the energy usage as well as the total task completion time. In addition, each robot can obtain partial knowledge of tasks and robots in the environment; they can then decide whether to stay and complete tasks they know or roam around for other tasks. In [Bar+20], more cost factors are added to the objective function. In addition to minimizing the total traveling distance, the authors aim to balance the load between robots as well as to maximize a quality satisfaction. Each task has a level of quality requirement, and so do the robots. To be considered satisfied, the difference in quality metric between the assigning task and the robot has to be minimized.

1.5.2 Optimization-based

Optimization has been used in many various fields. It aims to find an optimal or close to optimal solution within a set of constraints by minimizing a cost or maximizing a profit of a system [KHE15]. There exists a huge collection of optimization-based approaches for

MRTA depending on the nature of the problem.

When the allocation problem is small, with a few tasks and robots, it is possible to find the optimal solution using an exact search algorithm. Mixed-Integer Linear Programming is one of the exact algorithms, for which the allocation problem is formulated as integers and linear equations. In [SR15], this approach is used to solve the heterogeneous mTSP. The authors focus on having the salesman (robot) that starts from different points. Some of the targets are set to require specific robots to visit; this is set to reflect the heterogeneous nature of the robots. The algorithm aims to minimize the sum of all traveling costs, which are computed in a function of the euclidean distance between two points.

Another well-known exact search is the Hungarian algorithm. It is exploited in [Xue+21] to solve task allocation of unmanned surface vehicles, which have to move from their starting position to different target points. The goal is to minimize the max task time and reduce the total task time. The task time is a combination of the travel time and the turning time at initial and target points.

On the other hand, if the number of tasks and robots becomes too large, the exact solution approach might not be able to sustainably find the optimal solution in a finite amount of time. In this case, approximate search algorithms can be used. They may not find the general optimal solution, but an approximation that is close to the global one. Simulated Annealing (SA), a trajectory-based metaheuristic, is an approximation approach that consists of a single agent that traverses through the search space. In order to avoid being stuck in local minima, the SA accepts poor solutions within a certain probability during the search for the global optimum. An example of the use of SA can be found in [MM06]. The authors propose a solution to the mTSP using the *MinMax* cost function, which is based on the traveling cost between target points.

Another type of metaheuristic is population-based optimization, examples of which are Genetic Algorithm (GA), Particle Swarm Optimization (PSO), and Ant Colony Optimization (ACO). Inspired by genetic selection, the GA relies on crossover and mutation operations to find potential solutions, so-called chromosomes. A fitness function is introduced, against which the chromosomes are evaluated. The work in [Tsa+18] is one of the MRTA that implements this GA. The authors propose a fitness function that corresponds to the traveling distance of each task for the robot. Similar to GA, PSO relies on evolution over generations to search for optimal solutions. However, it does not have evolutionary operators such as crossover or mutation. Instead, PSO initializes a population, called particles, over a search space. Each particle updates its behavior based on its own trajec-

tory as well as the two best-found solutions: personal best and global best. One of the PSO work on MRTA can be found in [WJC20]. The authors propose a multi-objective PSO, whose cost function tries to minimize the total cost of all robots (*MinSum*) and the worst cost of a robot (*MinMax*) at the same time. They are both in the function of travel distance. Last but not least, the ACO introduces the concept of virtual ants. These ants start off to search for food (potential solution) leaving a trail behind, known as pheromone deposition. Other ants update their trajectories taking into account this trail. Over time, this trail can be evaporated leaving the path less attractive. In contrast, the more pheromone deposition there is, the more ants are likely to follow that trail; this induces positive feedback leading to eventually having all ants search on the same path.

1.6 Conclusion

In this chapter we have discussed the three different architectures of control used in the multi-robot system: centralized, distributed, and hierarchical. Even though they inherit different advantages and disadvantages, one does not have a clear edge over another. It depends on the actual MRS application considering the environment, number of robots, number of missions, etc.

The formation is a crucial part of any cooperative transportation task. Apart from the leader-follower, virtual structure, and behavior-based, which are the most well-known approaches, there exists also studies that are based on optimization and machine learning. However, not all of them are applied to the transportation mission. In addition to conventional formation control, which includes forming a shape and navigation, extra studies have to be conducted for such tasks; for instance, the load handling or carrying techniques. We have detailed three common strategies: pushing-only, caging, and grasping; they can have an impact on how the formation control method is chosen.

Since our goal is to transport the load directly on top of the robots, a fixed inter-distance formation control is required. In this thesis, we base the formation on two different control strategies: the consensus algorithm and the hierarchical quadratic programming approach. We consider using the consensus control because it allows the formation problem to be resolved in a distributed manner, thus increasing the scalability of the system. On the other hand, the hierarchical quadratic programming is a more relevant solution considering our use cases; it allows the formation control to clearly specify priorities for different behaviors. For instance, we can prioritize having a fixed inter-distance

formation over team navigation.

Last but not least, we have presented an introduction to the multi-robot task allocation. The type of MRTA is divided into three poles of taxonomy. Since MRTA is a challenge of optimization, various objective cost functions can be formulated. Then a potential optimal solution can be found using widely-known strategies such as auction-based and optimization-based. There is a huge list of MRTA solving strategies that fall into these two categories. These strategies are tailored to specific requirements and assumptions of various applications. However, few of the proposed approaches are validated in an actual real-world environment, and many of them are validated only in simulations.

In this work, the Contract Net Protocol and the Tabu Search are chosen. The CNP represents the auction-based approach, which can solve the allocation problem in a distributed way. However, its solution can be less optimal compared to its optimization-based counterpart. Hence, Tabu Search is our next choice representing the optimization-based allocation because it allows the solver to step a bit out of a minimum in order to find better minima values. These two approaches are then implemented and compared considering our use cases.

Chapter 2

Formation control

Contents

2.1 Introduction

The aim of this chapter is to present two proposed formation strategies that are based on different control architectures. The consensus-based formation is a distributed controller, whereas the optimization-based formation is a centralized approach. The consensus provides a solution to the formation problem by allowing each robot to react and compute its control velocity locally based on its neighbors. On the other hand, the optimization approach, which is based on the HQP, allows us to set behavior priorities. This results in having a more stable and fixed inter-distance formation.

We will discuss some fundamental mathematical theories that build up to the proposed consensus algorithm. These include graph theory and the consensus theory itself. The later half of this chapter is dedicated to the optimization-based controller. In that section, we introduce the hierarchical quadratic programming approach as well as definitions of sub-tasks for the optimization problem.

The two proposed control laws are validated in both simulations and real-world experiments. The results are presented in their respective controller sections. Our deployment objective for all robots in the team is to cooperatively transport a load. It requires constant inter-distances between them throughout navigation. This is the main attribute we are looking for.

2.2 Consensus-based formation

The consensus algorithm is a process of reaching an agreement on some values of a system. It is applied to a wide range of applications in the field of distributed computing and multi-agent systems. For robotics, consensus can be used as a control approach whose objective is to get to an agreement on some control features, for instance, the states of robots in the network [Hou+17]. Equation (2.1) is an example of the consensus problem; the goal is to maintain a relative position to the neighbors by minimizing a cost function:

$$\mathbf{v_i} = \sum_{j \in N_i} \|\mathbf{p_j} - \mathbf{p_i} - \mathbf{p_{ij}}\|^2 \tag{2.1}$$

where $\mathbf{v_i}$ is the control input of robot i, $\mathbf{p_{ij}}$ is a desired relative position of robot i and its neighbor j, and $\mathbf{p_i}$ and $\mathbf{p_j}$ is the position of robot i and j, respectively. If $\mathbf{p_{ij}} = \begin{pmatrix} 0 & 0 \end{pmatrix}^T$, all robots in the group will converge to a single point. Indeed, this relative position can be defined such that a desired shape of formation is obtained. From this simple technique, researchers have been pushing the boundary by introducing weight coefficients and/or designing more complex consensus terms; [Fal+11] and [Wu+19] are some examples of that.

Our consensus-based formation interest is to adapt the distributed flocking algorithm, which is introduced for the double integrator system in [SFZ19], for a single integrator formation system. We also introduce an improved obstacle avoidance, which is inspired by the work of [Olf06]. It considers each obstacle as a virtual robot, then a potential function is used to repel robots from the virtual robots (obstacles). The improvement to the

original obstacle avoidance approach is that each robot takes into account its neighbors' obstacle(s) in addition to its own obstacle(s), which is more suitable for preserving the formation compared to the original approach, where the obstacle avoidance is performed independently for each robot. This controller is then applied to a group of wheeled mobile robots, thus requiring a state linearization to overcome the nonholonomic constraint.

2.2.1 Wheeled mobile robots modeling

Let's consider a multi-robot system with n robots. Each robot is a differential drive robot, where $(x_{ri}, y_{ri})^T \in \mathbb{R}^2$ is the center of the i-th robot; θ_i, v_i and ω_i are its heading angle, linear velocity and angular velocity, respectively as shown in Figure 2.1. The kinematic model of this i-th robot can be expressed as follows:

$$\begin{pmatrix} \dot{x}_{ri} \\ \dot{y}_{ri} \\ \dot{\theta}_i \end{pmatrix} = \begin{pmatrix} \cos\theta_i & 0 \\ \sin\theta_i & 0 \\ 0 & 1 \end{pmatrix} \begin{pmatrix} v_i \\ \omega_i \end{pmatrix} \tag{2.2}$$

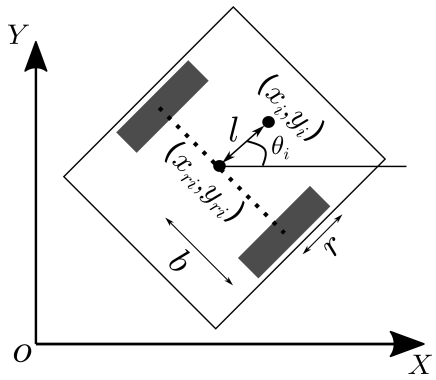


Figure 2.1 – A nonholonomic wheeled mobile robot.

The robot is subjected to nonholonomic constraints. Therefore, a static feedback method is used in order to apply any control law to the robots. An offset point, $\mathbf{p_i} = \begin{pmatrix} x_i & y_i \end{pmatrix}^T$ is defined from the robot's center as shown in Figure 2.1; as a result, the problem is shifted from controlling center of the robot to this offset point, which can be a point of interest later. Referenced to this point, the kinematic model is holonomic for $l \neq 0$, and it can be written as follows [RB08].

$$\mathbf{p_i} = \begin{pmatrix} x_i \\ y_i \end{pmatrix} = \begin{pmatrix} x_{ri} + l\cos\theta_i \\ y_{ri} + l\sin\theta_i \end{pmatrix} \tag{2.3}$$

Differentiating Equation (2.3) with respect to time, we can get the velocity of this offset in relation robot's linear and angular velocity as follows:

$$\dot{\mathbf{p}}_\mathbf{i} = \begin{pmatrix} \dot{x}_i \\ \dot{y}_i \end{pmatrix} = \mathbf{B}(\theta_i)\mathbf{u_i} \tag{2.4}$$

where $\mathbf{B}(\theta_i) = \begin{pmatrix} \cos\theta_i & -l\sin\theta_i \\ \sin\theta_i & l\cos\theta_i \end{pmatrix}$, $l \neq 0$ is the offset length, and $\mathbf{u_i} = \begin{pmatrix} u_i & \omega_i \end{pmatrix}^T$ is the velocity vector of i^{th} robot.

Expanding Equation (2.4) to n number of robots, the relation between velocities of each offset point and the robot is expressed as follows:

$$\mathbf{v} = \mathbf{J_x u} \tag{2.5}$$

where:

$$\mathbf{v} = \begin{pmatrix} \dot{\mathbf{p}}_\mathbf{1} \\ \dot{\mathbf{p}}_\mathbf{2} \\ \vdots \\ \dot{\mathbf{p}}_\mathbf{n} \end{pmatrix}, \quad \mathbf{J_x} = \begin{pmatrix} \mathbf{B}(\theta_1) & & \\ & \ddots & \mathbf{0} \\ \mathbf{0} & & \mathbf{B}(\theta_n) \end{pmatrix}, \quad \mathbf{u} = \begin{pmatrix} \mathbf{u_1} \\ \mathbf{u_2} \\ \vdots \\ \mathbf{u_n} \end{pmatrix}$$

The use of the offset point is extended to a mounting point between the load and group formation. Since this point is the control point of interest, it is practical to attach a load support to each robot at this point through a passive revolute joint. Figure 2.2 shows this idea of load placement. By mounting as such, the load's position and heading can be directly controlled since its frame is coincident with the formation frame.

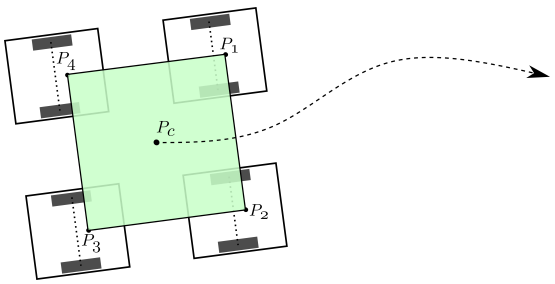


Figure 2.2 – Top view of the system. The green square represents a support for load placement that is on top of the formation.

Once the payload is mounted, the whole system can be seen as a single, highly overactuated mobile robot. Therefore, it is of prime importance to generate consistent control inputs for all robots such that the formation is kept. Ignoring these would lead to either set high forces on the passive joint or induce slipping or skidding of some robots. In both cases, the general motion of the formation would not be controlled correctly.

2.2.2 Graph theory

A graph is a pair of vertex set $\mathcal{V}$ and edge set $\mathcal{E}$ [Die05], $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ where $\mathcal{V} = \{1, 2 \ldots, n\}$ and $\mathcal{E} \subseteq \{(i, j) : i, j \in \mathcal{V}, j \neq i\}$. A vertex represents a node, an agent, or a robot, whereas an edge is formed by two connected vertices. Thus, each element of $\mathcal{E}$ is a pair of $\mathcal{V}$. Figure 2.3 shows an example of the graph consisting of five vertices $\mathcal{V} = \{1, 3, \ldots, 5\}$ and three edges $\mathcal{E} = \{\{1, 2\}, \{1, 3\}, \{4, 5\}\}$. The adjacency relationships of the such graph can be expressed in a matrix as follows:

$$\mathcal{A}(\mathcal{G}) = \begin{pmatrix} 0 & 1 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 \end{pmatrix} \tag{2.6}$$

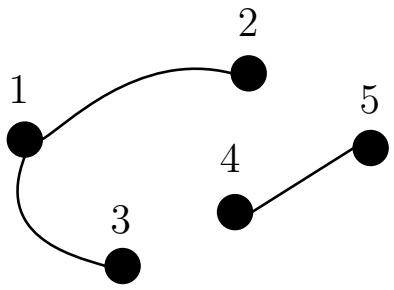


Figure 2.3 – Example of a graph.

The adjacency matrix, $\mathcal{A}(\mathcal{G})$, is a symmetric matrix with the dimension of $n \times n$, where n is the number of vertices, that represents the connectivity between neighbors:

$$[\mathcal{A}(\mathcal{G})]_{ij} = \begin{cases} 1 \text{ if } (i, j) \in \mathcal{E}, \\ 0 \text{ otherwise.} \end{cases} \tag{2.7}$$

In a multi-robot system, graph theory is often used to describe its topology. A topology is a description of physical relations between each robot in the group. It can be expressed by either an undirected or directed graph depending on the relations between neighbors. Vertices are symmetrically linked for the undirected graph, whereas the links are asymmetric for the directed graph. The edges indicate a two-way relationship in the undirected graph. For the directed graph, it's a one-way relationship; a robot can obtain information about a neighbor by explicitly communicating with that neighboring robot or by using its onboard sensor(s) to derive the information without actual communication. An example of the undirected and directed graph is illustrated in Figure 2.4a and Figure 2.4b, respectively. Notice that each link (edge) between vertices is represented by an oriented arrow for the directed graph in order to indicate a constraint motion of an agent with respect to another. For instance, in Figure 2.4b the arrow's direction flows from vertex (robot) 1 to vertex (robot) 2 indicating that the motion of robot 1 is constrained to robot 2. On the other hand, the robot 2 is unaware of 1, and its motion is unconstrained.

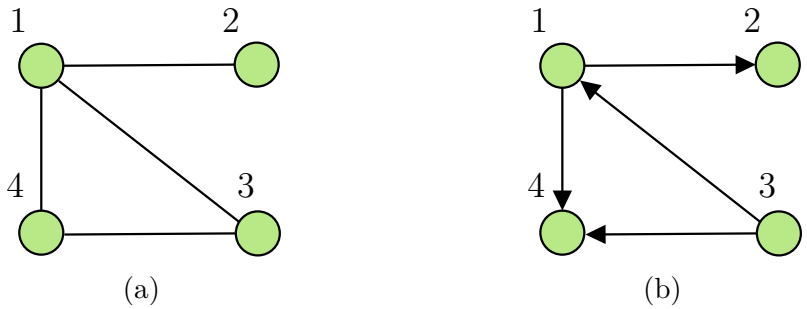


Figure 2.4 – Example of (a) undirected and (b) directed graph.

A graph is considered to be *connected* if there is a path that is linked between any pair of vertices. Otherwise, the graph is defined as *disconnected*. For the case of undirected graph, the connectivity can only be either connected (Figure 2.5a) or disconnected (Figure 2.5d). The directed graph, however, can be further classified into weakly (Figure 2.5b) and strongly (Figure 2.5c) connected graphs depending on if there exists a directed path between each vertex pair or not.

Another crucial aspect of the graph theory in formation is rigidity. A rigid graph is mainly applicable to an undirected graph, where the formation's shape is maintained throughout the maneuver [Ren+10]. For instance, if the formation is controlled by inter-distances between robots, these distances will have to remain constant at their corresponding desired values. In a rigid graph, the number of edges can grow significantly with

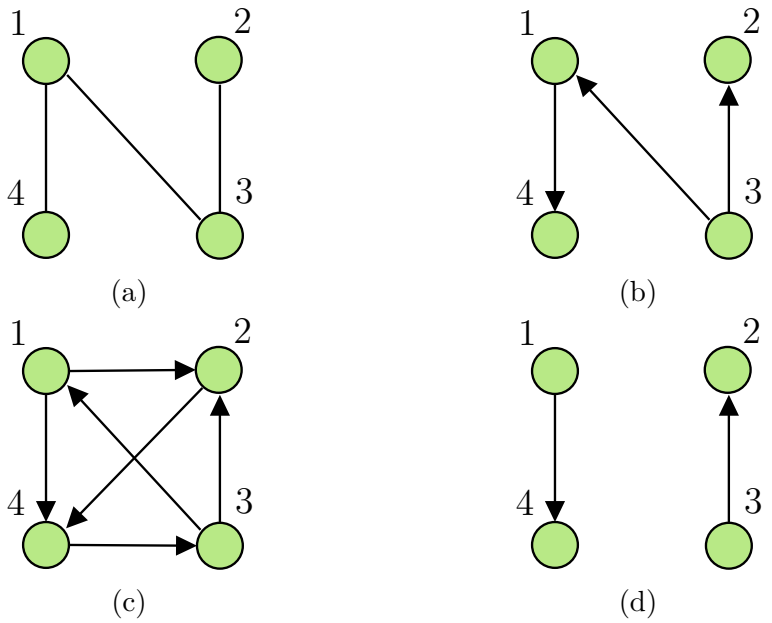


Figure 2.5 – Example of the connectivity of a graph: (a) connnected, (b) weakly connected, (c) strongly connected, and (d) disconnected graph.

respect to the number of vertices or robots. In fact, for a fully-connected graph, the total number of edges is:

$$\frac{n(n-1)}{2} \tag{2.8}$$

where n is the number of robots in the formation. Therefore, a concept of *minimally rigid graph* is commonly used instead. Minimally rigid graph corresponds to a graph where no edge can be further removed without losing its rigidity, hence the word minimal [FH08]. There exists also a more constrained version of rigidity, named infinitesimally rigid. It is when the vertices are allowed to move infinitesimally while still preserving the rigidity. To achieve this, the motion has to satisfy Equation (2.9) [Zel+15].

$$(\mathbf{v_j} - \mathbf{v_i})^T(\mathbf{p_j} - \mathbf{p_i}) = 0,\ \forall(i,j) \in \mathcal{E} \tag{2.9}$$

For a two-dimensional system, the minimal rigidity can be obtained when the total number of edges in the graph is:

$$2n - 3 \tag{2.10}$$

where each subgraph of n' vertices, where $n' \leq n$, has no more than $2n' - 3$ edges. The

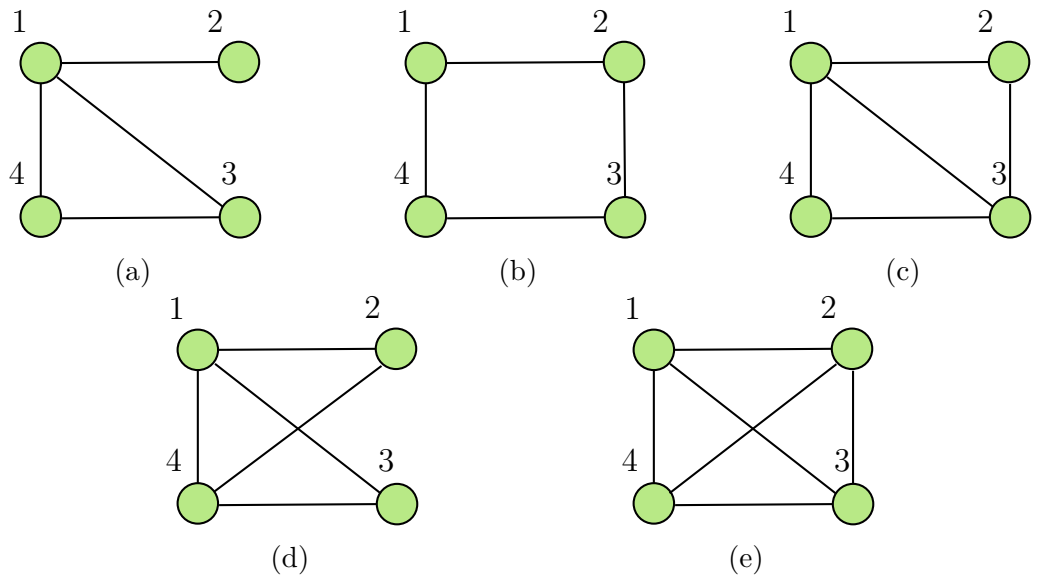


Figure 2.6 – Example of a graph rigidity in 2D: (a) and (b) flexible/non-rigid, (c) and (d) minimally rigid, and (e) rigid.

configuration of a minimally rigid graph is, therefore, not unique as shown in Figure 2.6c and Figure 2.6d. Figure 2.6 illustrates an example of the rigidity characteristic. The graph Figure 2.6a and Figure 2.6b are flexible or non-rigid as the vertex 2 of Figure 2.6a can swing freely with a fixed radius around vertex 1, whereas in Figure 2.6b, the shape can change to a parallelogram without affect the distances between robots. On the other hand, one can remove any edge from the graph in Figure 2.6e without rendering it flexible.

Our system is represented by an undirected graph. Each robot is denoted by a node of the vertices, and the set of neighbors of node i, N_i, is defined by:

$$N_i = \{j \in \mathcal{V} : (i,j) \in \mathcal{E}\} = \{j \in \mathcal{V} : \|\mathbf{p_j} - \mathbf{p_i}\| < c\} \tag{2.11}$$

where $c > 0$ is the interaction range between robots, and $\|.\|$ denotes the Euclidean norm. $\mathbf{p_i}$ and $\mathbf{p_j}$ are the Cartesian coordinates of robots i and j, respectively.

The desired formation of the system can be expressed as:

$$\|\mathbf{p_j} - \mathbf{p_i}\| = d \quad \forall j \in N_i(\mathbf{p}) \tag{2.12}$$

where d is the desired inter-distance between robot i and j.

2.2.3 Preliminaries

This section discusses the flocking control for the double-integrator system. We start off by introducing the preliminaries and the control algorithm proposed by [Olf06], followed by the modified PID-like controller of [SFZ19].

Similar to the adjacency matrix of the topological connection in the graph theory, a spatial adjacency matrix $\mathcal{A}(\mathbf{p}) = [a_{ij}(\mathbf{p})]$; it is defined in order to reflect on the position-based connection between robots. The elements a_{ij} of this matrix are given as follows:

$$a_{ij}(\mathbf{p}) = \begin{cases} 0 & \text{if } i = j \\ \rho_h(\|\mathbf{p_j} - \mathbf{p_i}\|_\sigma / \|c\|_\sigma) & \text{if } i \neq j \end{cases} \tag{2.13}$$

where p_h is called a bump function and $\|.\|_\sigma$ is called σ-norm. The bump function is a smooth scalar function ranging between zero and one: $\mathbb{R}^+ \to [0, 1]$. With $h \in (0, 1)$, this function is defined as follows:

$$\rho_h(z) = \begin{cases} 1, & z \in [0, h) \\ \frac{1}{2}\left[1 + cos\left(\pi \frac{(z-h)}{(1-h)}\right)\right], & z \in [h, 1] \\ 0, & \text{otherwise} \end{cases} \tag{2.14}$$

The σ-norm is a map of a vector to positive value: $\mathbb{R}^n \to \mathbb{R}^+$, $n \in \mathbb{Z}^+$. This norm is defined as follows:

$$\|\mathbf{z}\|_\sigma = \frac{1}{\epsilon}\left[\sqrt{1 + \epsilon\|\mathbf{z}\|^2} - 1\right] \tag{2.15}$$

with $\epsilon > 0$, and its gradient can be expressed as:

$$\sigma_\epsilon(\mathbf{z}) = \nabla\|\mathbf{z}\|_\sigma = \frac{\mathbf{z}}{\sqrt{1 + \epsilon\|\mathbf{z}\|^2}} = \frac{\mathbf{z}}{1 + \epsilon\|\mathbf{z}\|_\sigma} \tag{2.16}$$

Thus unlike $\|\mathbf{z}\|$, which is not differentiable at $\mathbf{z} = 0$, $\|\mathbf{z}\|_\sigma$ is differentiable everywhere.

A smooth pairwise attractive/repulsive potential function is defined as:

$$\Psi_\alpha(z) = \int_{\|d\|_\sigma}^{z} \Phi_\alpha(\delta) d\delta \tag{2.17}$$

This function is an integration of an *action* function Φ_α that has a minimum at $z = \|d\|_\sigma$ and a finite cut-off when $z \geq \|c\|_\sigma$. An example of this potential function $\Psi_\alpha(z)$ is depicted

in Figure 2.7. The action function is defined as follows:

$$\begin{aligned} \Phi_\alpha(z) &= \rho_h(z/\|c\|_\sigma)\Phi(z - \|d\|_\sigma) \\ \Phi(z) &= \frac{1}{2}\left[(a+b)\sigma_1(z+e) + (a-b)\right] \end{aligned} \tag{2.18}$$

where $\sigma_1(z) = z/\sqrt{1+z^2}$. With $0 < a \leq b$ and $e = |a-b|/\sqrt{4ab}$, the uneven sigmoidal function $\Phi(z) = 0$ when $z = 0$.

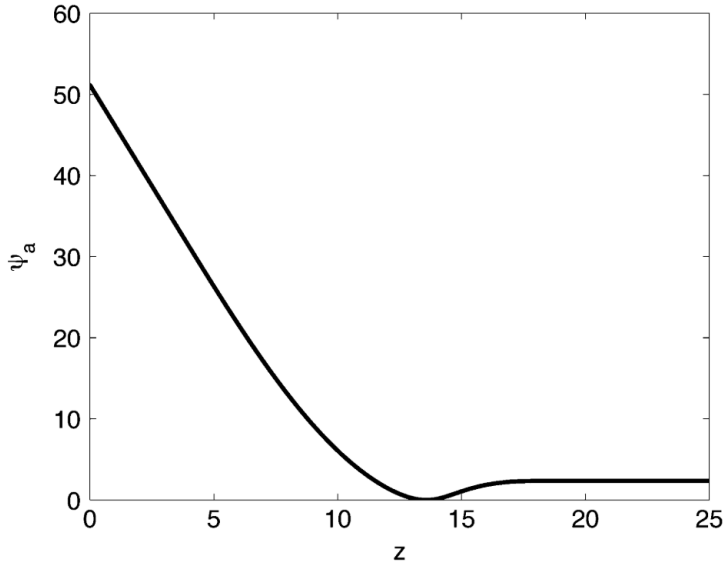


Figure 2.7 – Smooth pairwise function $\Psi_\alpha(z)$ [Olf06].

Therefore, a smooth collective potential function is introduced as:

$$V(\mathbf{p}) = \frac{1}{2}\sum_i \sum_{j \neq i} \Psi_\alpha\left(\|\mathbf{p_j} - \mathbf{p_i}\|_\alpha\right) \tag{2.19}$$

With the above attractive/repulsive characteristic, a control law introduced by [Olf06], which applied on the second-order dynamic of each robot, is defined as follows:

$$\dot{\mathbf{v}}_\mathbf{i} = \sum_{j \in N_i} \left[\Phi_\alpha(\|\mathbf{p_j} - \mathbf{p_i}\|_\sigma)\mathbf{n_{ij}} + a_{ij}(\mathbf{p})(\mathbf{v_j} - \mathbf{v_i})\right] + \mathbf{f_i}(\mathbf{v_i}, \mathbf{p_i}, \mathbf{v_r}, \mathbf{p_r}) \tag{2.20}$$

where $\mathbf{f_i}(\mathbf{v_i}, \mathbf{p_i}, \mathbf{v_r}, \mathbf{p_r}) = -c_1(\mathbf{p_i} - \mathbf{p_r}) - c_2(\mathbf{v_i} - \mathbf{v_r}), c_1, c_2 > 0$ and $\mathbf{n_{ij}} = \sigma_\epsilon(\mathbf{p_j} - \mathbf{p_i})$.

This control law is made up of three terms: 1) a gradient-based term that regulates the inter-distance between robots, 2) a velocity consensus for flocking, and 3) a navigational feedback control, where $\mathbf{p_r}$ and $\mathbf{p_r}$ are the desired position and velocity to be tracked,

respectively.

To further improve the system's behavior, a PID-like tunable gains version of this control algorithm is introduced by [SFZ19]. It is given as follows:

$$\begin{aligned}\dot{\mathbf{v}}_\mathbf{i} = \sum_{j\in N_i} [K_p \Phi_\alpha(\|\mathbf{p_j} - \mathbf{p_i}\|_\sigma)\mathbf{n_{ij}} + K'_p a_{ij}(\mathbf{p})(\mathbf{p_j} - \mathbf{p_i}) + K_d a_{ij}(\mathbf{p})(\mathbf{v_j} - \mathbf{v_i}) \\ + K_i \int \sum_{j\in N_i} \Phi_\alpha(\|\mathbf{p_j} - \mathbf{p_i}\|_\sigma)\mathbf{n_{ij}} dt] + \mathbf{f_i}(\mathbf{v_i}, \mathbf{p_i}, \mathbf{v_r}, \mathbf{p_r})\end{aligned} \tag{2.21}$$

where $K_p, K'_p, K_d, K_i > 0$ are the tuning gains of the controller. The authors' perspective of introducing K_p, K'_p, K_d is to compensate for the uncertainties when applying to a nonlinear system, whereas the integral term with K_i is added in order to overcome the steady-state error problem.

Since this flocking algorithm does not include any obstacle avoidance capability in the control law, we incorporate the obstacle avoidance algorithm from [Olf06].

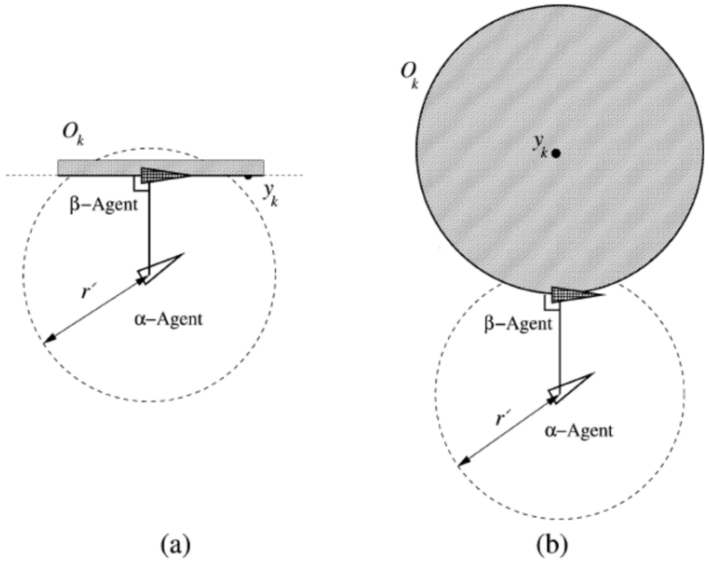


Figure 2.8 – Obstacles denoted by agent-based approach: (a) wall and (b) spherical obstacles [Olf06].

Figure 2.8 shows the agent-based approach of representing obstacles. An α-agent represents actual robots while a β-agent indicates the obstacles. The set of neighbors of node i can be written as:

$$\begin{aligned} N_i^\alpha &= j \in \mathcal{V}_\alpha : \|\mathbf{p_j} - \mathbf{p_i}\| < c \\ N_i^\beta &= k \in \mathcal{V}_\beta : \|\widehat{\mathbf{p}}_{\mathbf{i,k}} - \mathbf{p_i}\| < c' \end{aligned} \tag{2.22}$$

Incorporating the agent-to-obstacle constraint into the control law, the overall constraints of the system, thus, consist of:

$$\begin{cases} \|\mathbf{p_j} - \mathbf{p_i}\| = d & \forall j \in N_i^\alpha \\ \|\widehat{\mathbf{p}}_{\mathbf{i,k}} - \mathbf{p_i}\| = d' & \forall k \in N_i^\beta \end{cases} \tag{2.23}$$

where c' and d' are the interaction range and inter-distance between α-agent i and β-agent k, respectively. $\widehat{\mathbf{p}}_{\mathbf{i,k}}$ is an estimated position, and $\widehat{\mathbf{v}}_{\mathbf{i,k}}$ is an estimated velocity of the closest point from robot i to obstacle k. These estimations of position and velocity can be done using either external or onboard sensors such as lidar and camera. Since obstacles are treated as virtual robots, the avoidance control law is similar to the flocking control law based on a potential function, which is defined as follows:

$$\dot{\mathbf{v}}_\mathbf{i}^\beta = c_1^\beta \sum_{k \in N_i^\beta} \Phi_\beta(\|\widehat{\mathbf{p}}_{\mathbf{i,k}} - \mathbf{p_i}\|_\sigma)\widehat{\mathbf{n}}_{\mathbf{i,k}} + c_2^\beta \sum_{k \in N_i^\beta} b_{i,k}(\mathbf{p})(\widehat{\mathbf{v}}_{\mathbf{i,k}} - \mathbf{v_i}) \tag{2.24}$$

where $c_1^\beta, c_2^\beta > 0$ are gains, $\widehat{\mathbf{n}}_{\mathbf{i,k}} = \sigma_\epsilon(\widehat{\mathbf{p}}_{\mathbf{i,k}} - \mathbf{p_i})$, and $b_{i,k}(\mathbf{p})$, which is the heterogeneous adjacency between an α-agent i and its neighboring obstacle k at $\widehat{\mathbf{p}}_{\mathbf{i,k}}$, is defined as:

$$b_{i,k}(\mathbf{p}) = \rho_h\left(\|\widehat{\mathbf{p}}_{\mathbf{i,k}} - \mathbf{p_i}\|_\sigma / \|d'\|_\sigma\right) \tag{2.25}$$

with the function defining the repulsive action expressed as follows:

$$\Phi_\beta(z) = \rho_h(z/\|d'\|_\sigma)(\sigma_1(z - \|d'\|_\sigma) - 1) \tag{2.26}$$

2.2.4 Formation control algorithm

If we combine the flocking control, Equation (2.21), with the obstacle avoidance, Equation (2.24), we get the overall control law as follows:

$$\begin{aligned}\dot{\mathbf{v}}_\mathbf{i} = &\sum_{j\in N_i^\alpha}[K_p\Phi_\alpha(\|\mathbf{p_j}-\mathbf{p_i}\|_\sigma)\mathbf{n_{ij}} + K_p' a_{ij}(\mathbf{p})(\mathbf{p_j}-\mathbf{p_i}) + K_d a_{ij}(\mathbf{p})(\mathbf{v_j}-\mathbf{v_i})] \\ &+ K_i\int\sum_{j\in N_i^\alpha}\Phi_\alpha(\|\mathbf{p_j}-\mathbf{p_i}\|_\sigma)\mathbf{n_{ij}}dt + \mathbf{f_i}(\mathbf{v_i},\mathbf{p_i},\mathbf{v_r},\mathbf{p_r}) \\ &+ c_1^\beta\sum_{k\in N_i^\beta}\Phi_\beta(\|\widehat{\mathbf{p}}_\mathbf{i,k}-\mathbf{p_i}\|_\sigma)\widehat{\mathbf{n}}_\mathbf{i,k} + c_2^\beta\sum_{k\in N_i^\beta} b_{i,k}(\mathbf{p})(\widehat{\mathbf{v}}_\mathbf{i,k}-\mathbf{v_i})\end{aligned} \tag{2.27}$$

Since our goal is to apply a formation control to a single-integrator system, we are interested only in the gradient-based term for regulating the inter-distance and not so much in the velocity matching term. The overall control law, Equation (2.27), is thus modified to be as follows:

$$\begin{aligned}\mathbf{v_i} = &\sum_{j\in N_i^\alpha}K_p\Phi_\alpha(\|\mathbf{p_j}-\mathbf{p_i}\|_\sigma)\mathbf{n_{ij}} + K_i\int\sum_{j\in N_i^\alpha}\Phi_\alpha(\|\mathbf{p_j}-\mathbf{p_i}\|_\sigma)\mathbf{n_{ij}}dt \\ &- \mathrm{sat}(c_1^\gamma(\mathbf{p_i}-\mathbf{p_r})) + \mathbf{v_i^\beta}\end{aligned} \tag{2.28}$$

where $\mathbf{v_i^\beta} = c_1^\beta\sum_{k\in N_i^\beta}\Phi_\beta(\|\widehat{\mathbf{p}}_\mathbf{i,k}-\mathbf{p_i}\|_\sigma)\widehat{\mathbf{n}}_\mathbf{i,k}$ and sat(.) is a saturation function defined as:

$$\mathrm{sat}(x) = \begin{cases} x_{min}, & x < x_{min} \\ x_{max}, & x > x_{max} \\ x, & \text{otherwise.}\end{cases} \tag{2.29}$$

The value of x_{min} and x_{max} are chosen explicitly in order to ensure that the navigation term will not dominate over other control terms when the goal point is too far away.

Moreover, in order to ensure the formation shape while navigating around the obstacle, we propose a new consensus obstacle avoidance scheme. In this approach, each robot will take into account its neighbors' obstacles as well. By doing so, it can anticipate incoming obstacles even if they are not yet in its own detection range. Hence, the robots in the group will synchronously avoid the obstacle. The obstacle avoidance term of Equation (2.28), $\mathbf{v_i^\beta}$ now takes into account all potential obstacles for all robots, which is expressed as follows:

$$\mathbf{v_i^\beta} = c_1^\beta\sum_{k\in N_i^\beta}\Phi_\beta(\|\widehat{\mathbf{p}}_\mathbf{i,k}-\mathbf{p_i}\|_\sigma)\widehat{\mathbf{n}}_\mathbf{i,k} + \sum_{j\in N_i^\alpha}\sum_{k\in N_j^\beta}\Phi_\beta(\|\widehat{\mathbf{p}}_\mathbf{j,k}-\mathbf{p_j}\|_\sigma)\widehat{\mathbf{n}}_\mathbf{j,k} \tag{2.30}$$

Combining Equation (2.28) and (2.30), the overall proposed control law for a single integrator system is as follows:

$$\begin{aligned}\mathbf{v_i} = &\sum_{j\in N_i^\alpha} K_p \Phi_\alpha(\|\mathbf{p_j}-\mathbf{p_i}\|_\sigma)\mathbf{n_{ij}} + K_i \int \sum_{j\in N_i^\alpha} \Phi_\alpha(\|\mathbf{p_j}-\mathbf{p_i}\|_\sigma)\mathbf{n_{ij}}dt \\ &+ c_1^\beta \sum_{k\in N_i^\beta} \Phi_\beta(\|\widehat{\mathbf{p}}_{\mathbf{i,k}}-\mathbf{p_i}\|_\sigma)\widehat{\mathbf{n}}_{\mathbf{i,k}} + \sum_{j\in N_i^\alpha}\sum_{k\in N_j^\beta} \Phi_\beta(\|\widehat{\mathbf{p}}_{\mathbf{j,k}}-\mathbf{p_j}\|_\sigma)\widehat{\mathbf{n}}_{\mathbf{j,k}} \\ &- sat(c_1^\gamma(\mathbf{p_i}-\mathbf{p_r}))\end{aligned} \tag{2.31}$$

2.2.5 Simulations and experimental results

We validate the proposed control law in simulation as well as in the real experiment. In order to apply the control algorithm to a mobile robot, we use the static state feedback as explained in Section 2.2.1, specifically using Equation (2.4). In both simulation and experiment, ROS [1] is used as the middleware. In the case of simulation, we simulate the control laws from Equation (2.28) and (2.30). For the experiment, we utilize four Huskys [2]. The control law from Equation (2.30) is run in each Husky independently, only the obstacle and its own positions, which are estimated using onboard sensors, in a known map are shared within a local network. The aim is to obtain a formation control that can keep its shape throughout the navigation, thus the inter-distances between robots should be constant.

The value of parameters used in the simulation and experiment are shown in Table 2.1. The interaction range, c, for control law from Equation (2.28) is chosen according to the remark of authors in [Olf06], while for Equation (2.30), it is defined such that each robot becomes a neighbor of all others in the team; in another word, a fully-connected graph. The minimum and maximum values of the saturation function are -0.3 and 0.3, respectively.

Simulation

Figure 2.9 illustrates the simulation architecture used. Gazebo [3] is used to simulate four robots under one ROS master. Each robot has its own localization, obstacle detection, and formation control node.

The simulated environment is shown in Figure 2.10. The obstacle is presented as the red cylinder whereas the green one represents the goal point.

1. https://www.ros.org
2. https://e-cobot.com/husky-robot-mobile-intelligent-2/
3. https://gazebosim.org

Table 2.1 – Value of parameters for the consensus control law.

Parameter	Value	Parameter	value [m]
a	1.0	x_{c_g}	8.0
b	1.0	y_{c_g}	-3.0
h	0.2	d	2.0
ϵ	0.1	d'	1.5
K_p	0.2	c for (2.28)	2.5
K_i	0.05	c for (2.30)	5.0
c_1^β	0.3	l	0.2
c_1^γ	0.25		

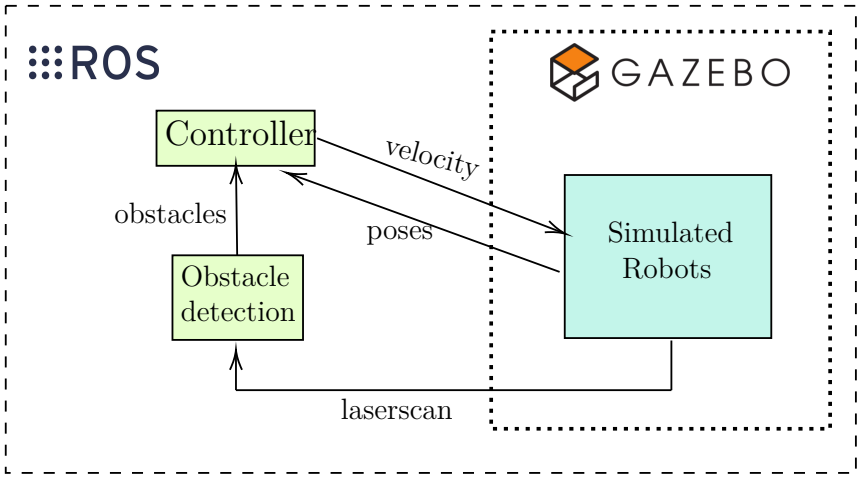


Figure 2.9 – Simulated system architecture.

A simulation of Equation (2.28) is shown in Figure 2.11. It can be seen in Figure 2.11b that robot 0 (red) breaks formation in order to avoid the obstacle. More precisely, this robot avoids the obstacle from the left side while the other robots avoid it from the right side. Indeed, in Figure 2.11a it was clear that the inter-distances took a huge amount of time to stabilize. The formation shape is totally distorted and restored only after the obstacle is passed through. This is, indeed, a common behavior as shown in [Olf06]. This behavior is not desirable considering we want the formation to be kept always, which is important for the application of load transportation.

The result of the proposed control law, Equation (2.30), is shown in Figure 2.12. In Figure 2.12a, d_{ij} represents the inter-distance between robots i and j, where $i, j \in$

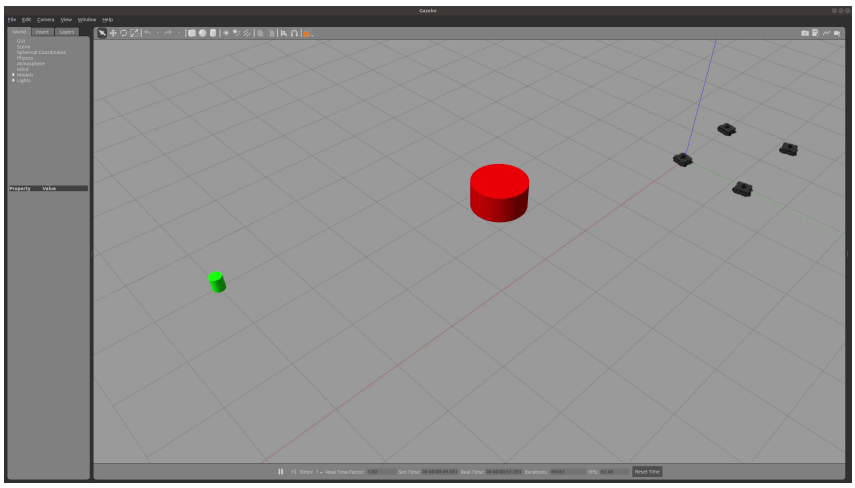
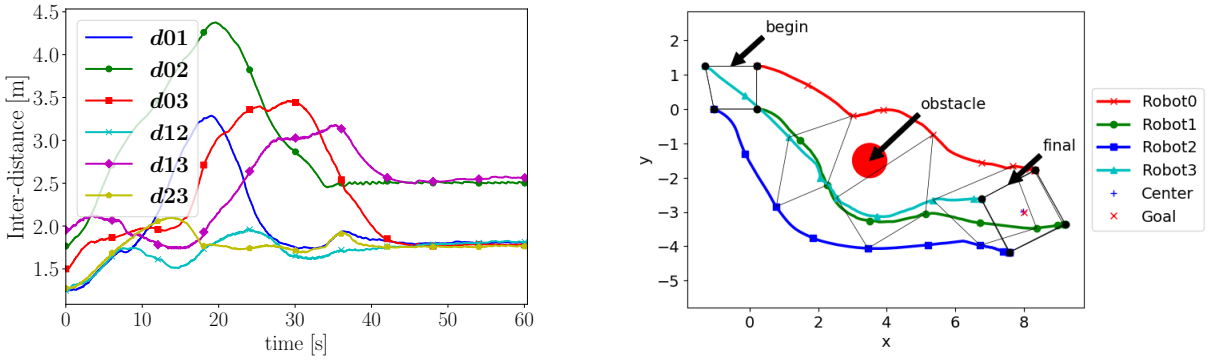


Figure 2.10 – Gazebo environment for simulation.



(a) Inter-distances between robots.

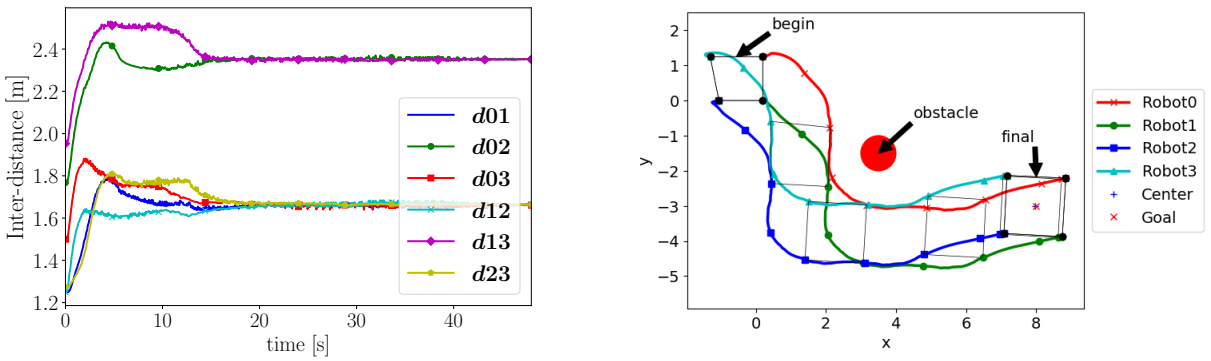
(b) Robots' trajectories.

Figure 2.11 – Simulation of formation and obstacle avoidance using flocking / avoidance law of Equation (2.28).

$\{0, 1, 2, 3\}$ and $i \neq j$. The inter-distances between each robot quickly converge and stay stable over time. Moreover, the formation of the team is preserved even during obstacle avoidance motion. As shown in Figure 2.12b, the shape is fixed throughout the navigation. This behavior is desirable, and we will see the actual implementation of this control algorithm in the next section.

Experiment

For the actual experiment, four robots are used; they are all independent of each other in terms of controller and ROS master node. The team is fully connected, meaning each



(a) Inter-distances between robots.

(b) Robots' trajectories.

Figure 2.12 – Simulation of formation and obstacle avoidance using the proposed control law, Equation (2.30).

robot broadcasts its own pose and obstacle position in a fixed reference frame to every other robot in the group. Obstacles that are not already mapped in the environment will be detected and avoided in real-time.

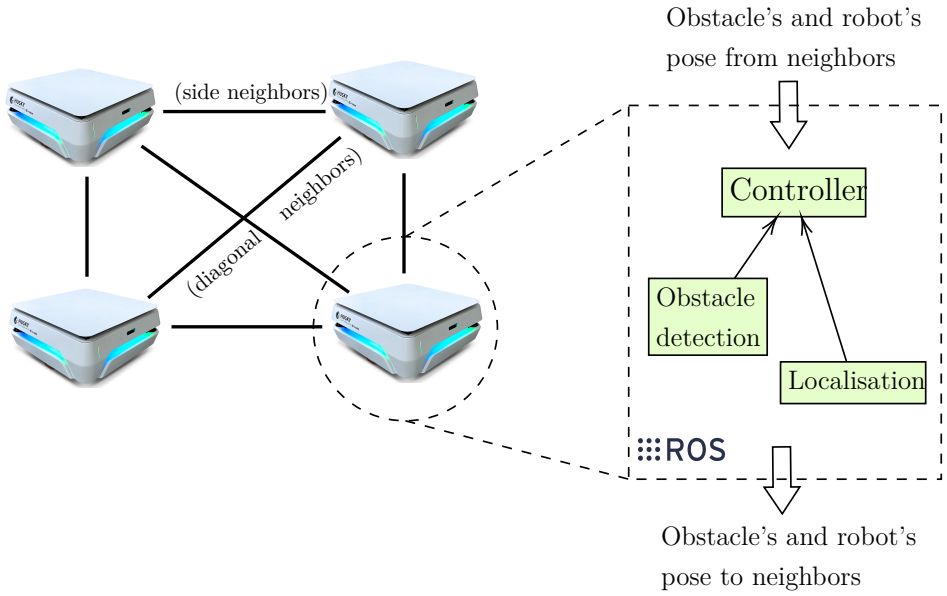
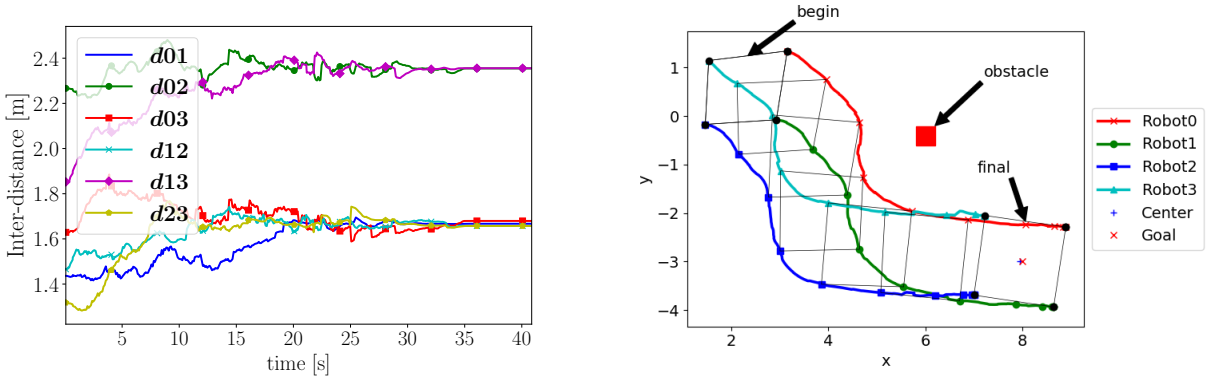


Figure 2.13 – Experimental setup of the consensus formation.

Figure 2.14 shows the implementation result of the proposed control law. From Figure 2.14a, we can notice the convergence of the inter-distances between robots. In fact, once the inter-distances start to converge and stabilize, the formation shape is kept regardless of the disturbance from an obstacle. Figure 2.14b shows the trajectories of the four

Huskys. Similar to simulation, the formation does not break while the team maneuvers away from the obstacle satisfying the requirements for cooperative transportation.



(a) Inter-distances between robots.

(b) Robots' trajectories.

Figure 2.14 – Real experiment of formation and obstacle avoidance using proposed control law, Equation (2.30).

Figure 2.15 shows a sequence of snapshots of this experiment, where the fleet has to move from one point to another. An obstacle (a carton box) is placed in between the two points. This obstacle is not already known in the map, hence no initial path planning can try to avoid it; it has to be detected and avoided by the whole fleet during the navigation.

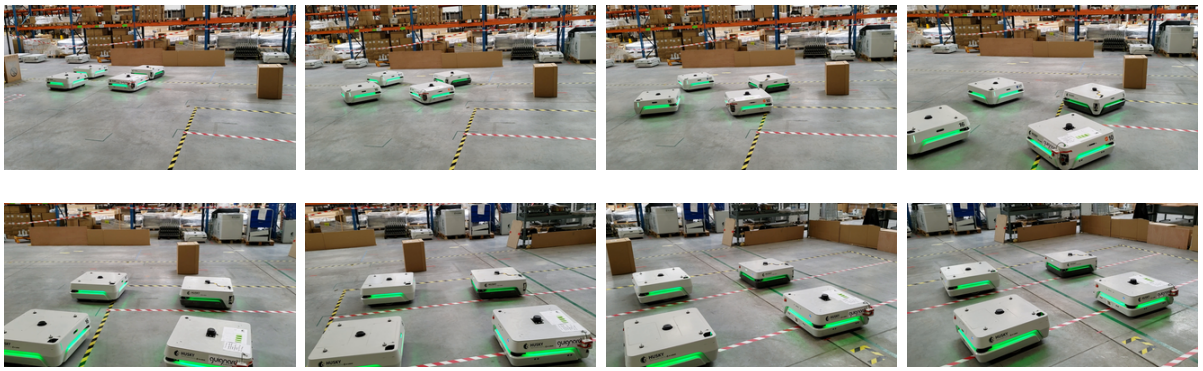
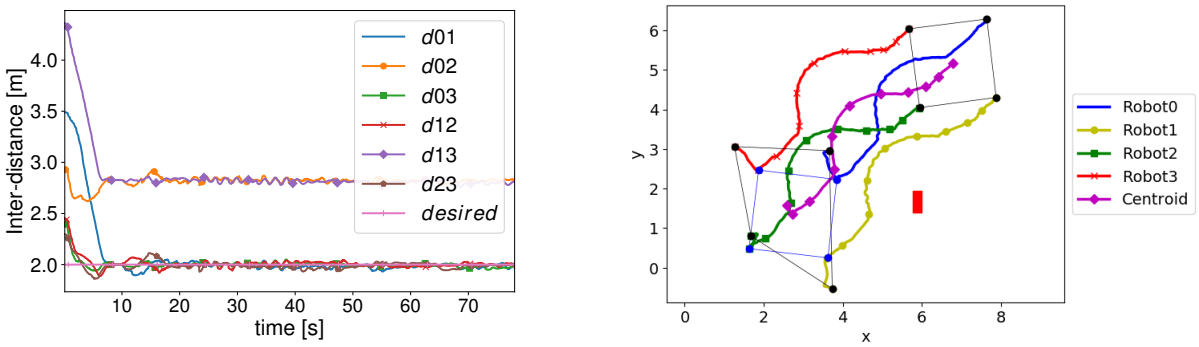


Figure 2.15 – Snapshots of the experiment.

Notice that the resultant inter-distance between robots on each side of the square is not equal to the desired inter-distance; this is because this desired value was set to all neighbors in the team, including those that are diagonal to each other. Hence the consensus tries to reach a compromise between all robots. Therefore, in the following experiment, we introduce a different desired inter-distance value for the diagonal neighbor

robots, more specifically it will be a square root of two of that of the side neighbors; it is corresponding to the shape of a perfect square.



(a) Inter-distances between robots.

(b) Robots' trajectories.

Figure 2.16 – Experiment of the formation and obstacle avoidance using the proposed control law with square root diagonal neighbor distance.

The formation result can be seen in Figure 2.16. The inter-distances of side and diagonal neighbors converge quickly to the desired ones, and they remain around the desired values even when maneuvering around the obstacle. This is backed by the robots' trajectories; the square shape is obtained at around $t = 10$s, indicated as a blue square in Figure 2.16b.

2.2.6 Conclusion

In this section, we have revised existing flocking algorithms such that they can be applied in nonholonomic robots systems. A combined consensus control for formation and obstacle avoidance is proposed, and it is able to do formation and navigation of the fleet's barycenter to a goal point without any collision with the obstacle. The validation of this control law shows its effectiveness in simulations as well as real experiments using actual industrial robots, Huskys. A video of these experiments can be found here [4].

However, the nature of this approach is a compromised control, which means the priority of each task (formation, navigation, obstacle avoidance, etc.) is not strictly enforced. This could potentially be a problem as the formation will not be kept if some parts of the controller dominate another. A solution to this issue is to introduce a strict

4. https://youtu.be/kS7HrOHnbho

hierarchy of tasks for the controller, which will be discussed in an optimization-based approach in the next section.

2.3 Optimization-based approach

In order to ensure the strict priority order of tasks, an optimization approach can be used, namely hierarchical quadratic programming (HQP). This section discusses the proposed formation controller, which is based on the HQP. Tasks are defined as either equality or inequality constraints with different levels of priority. These tasks include rigid shape formation control, group navigation, individual and team obstacle avoidance, and velocity limits.

2.3.1 Task-based control

Let us define $\mathbf{s} \in \mathbb{R}^{n_s}$ as a set of control features. The goal of the controller is to drive the value of these features to the desired ones, $\mathbf{s}^*$. In order to do this, a relation between the features' rate of changes, $\dot{\mathbf{s}}$, and the robots' velocities, $\mathbf{v}$, has to be determined, and it can be expressed as follows [KC14]:

$$\dot{\mathbf{s}} = \mathbf{J}_\mathbf{s}\mathbf{v} \tag{2.32}$$

where $\mathbf{J}$ is the Jacobian matrix that is, generally, as a function of the features.

As the goal is to regulate the feature error

$$\mathbf{e} = \mathbf{s} - \mathbf{s}^* \tag{2.33}$$

to zero, a proportional controller, $\dot{\mathbf{e}} = -\lambda\mathbf{e}$, can be used to ensure an exponential decay of $\mathbf{e}$ with λ is a positive gain. There the control input of the system, considering Equation (2.32), can be written as follows:

$$\mathbf{J}_\mathbf{s}\mathbf{v} = \dot{\mathbf{s}}^* - \lambda\mathbf{e} \tag{2.34}$$

where $\dot{\mathbf{s}}^*$ is assumed to be null as the set of desired values is fixed. This Equation (2.34) has a unique solution only if $\mathbf{J}_\mathbf{s}$ is a square and non-singular matrix, which is often not the case. Therefore, an approximation solution can be found using the following control laws:

$$\mathbf{v} = -\lambda \widehat{\mathbf{J}_\mathbf{s}^+} (\mathbf{s} - \mathbf{s}^*) \tag{2.35}$$

where $\widehat{\mathbf{J}_\mathbf{s}^+}$ is the pseudo-inverse of the estimation of the Jacobian.

2.3.2 Hierarchical quadratic programming

The hierarchical quadratic programming (HQP) framework [KLW11] is used to solve for control input ensuring strict priority order. HQP is a sequence of QP that tries to minimize a cost function subjected to equality and inequality constraints as well as solutions of higher priority tasks. At k-level of priority, HQP can be formulated as follows:

$$\begin{aligned} \mathbf{u_k} = \quad & \arg\min_\mathbf{x} && \|\mathbf{A_k x} - \mathbf{b_k}\|^2 \\ & \text{s.t.} && \mathbf{A_i x} = \mathbf{A_i u_i},\ \forall i < k \\ & && \mathbf{C_k x} \leq \mathbf{d_k} \\ & && \mathbf{C_i x} \leq \mathbf{d_i},\ \forall i < k \end{aligned} \tag{2.36}$$

where $\mathbf{u_k}$ is the optimal solution that minimizes the cost function of equality task k, $\mathbf{x}$ is the optimization variable, matrix $\mathbf{A_k}$ and vector $\mathbf{b_k}$ describe the cost function, whereas $\mathbf{A_i}$ and $\mathbf{b_i}$ represent the cost function of previous hierarchies. $\mathbf{C_k}$, $\mathbf{d_k}$ and $\mathbf{C_i}$, $\mathbf{d_i}$ are the inequality constraints of the current and previous levels, respectively. The equality constraint ensures that the objective task is minimized at best without disturbing previous higher priority tasks.

Note that the equality constraints inherited from previous hierarchy levels are actually $\|\mathbf{A_i x} - \mathbf{b_i}\|^2 = \|\mathbf{A_i u_i} - \mathbf{b_i}\|^2$. Writing them as in Equation (2.36) makes the constraints stronger but keeps them linear. The final solution can be chosen as the minimal-norm one by solving the last hierarchy level with the pseudo-inverse.

2.3.3 Task definitions

The tasks will be generally defined for n robots, and we chose specifically $n = 4$ for implementation examples. All the defined tasks are applied to the offsets' positions. We recall Equation (2.5) for the case of four robots, the relation between the velocity of the offset point and the control input of each robot in the case of four robots is expressed as follows:

$$\mathbf{v} = \mathbf{J_x u} \tag{2.37}$$

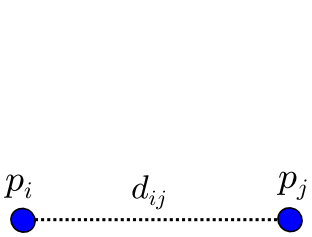
where:

$$\mathbf{v} = \begin{pmatrix} \dot{\mathbf{p}}_\mathbf{1} \\ \vdots \\ \dot{\mathbf{p}}_\mathbf{4} \end{pmatrix}, \mathbf{J_x} = \begin{pmatrix} \mathbf{B}(\theta_\mathbf{1}) & & \\ & \ddots & \mathbf{0} \\ \mathbf{0} & & \mathbf{B}(\theta_\mathbf{4}) \end{pmatrix}, \mathbf{u} = \begin{pmatrix} v_1 \\ \omega_1 \\ \vdots \\ v_4 \\ \omega_4 \end{pmatrix}$$

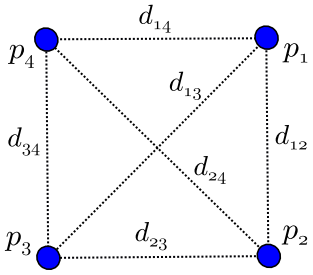
and $\dot{\mathbf{p}}_\mathbf{i}, i = 1, 2, 3, 4$, is defined as in Equation (2.4).

Geometric formation

The task of forming into a particular geometric shape is one of our specific goals mentioned in the previous Section 2.2. The inter-distances between each robot in the group are used as the control features. With user-defined values of the inter-distances, specific geometric shape formation can be done.



(a) Inter-distance between two robots ($\mathbf{p_i}$ and $\mathbf{p_j}$) denoted by d_{ij}.



(b) For four robots ($\mathbf{p_i}, i = 1, 2, 3, 4$).

Figure 2.17 – Inter-distances diagram of the MRS, each blue node represents a mobile robot in 2D plane.

If we consider two points as shown in Figure 2.17a with $\mathbf{p_i}, \mathbf{p_j} \in \mathbb{R}^2, i, j = 1, 2...$, the distance d_{ij} is as follows:

$$d_{ij} = \|\mathbf{p_j} - \mathbf{p_i}\| \tag{2.38}$$

By differentiating Equation (2.38) with respect to time, we can get the relation between the change of inter-distance and the change of robots' positions as follows:

$$\dot{d}_{ij} = \begin{pmatrix} -\dfrac{(\mathbf{p_j} - \mathbf{p_i})^T}{d_{ij}} & \dfrac{(\mathbf{p_j} - \mathbf{p_i})^T}{d_{ij}} \end{pmatrix} \begin{pmatrix} \dot{\mathbf{p}}_i \\ \dot{\mathbf{p}}_j \end{pmatrix} \tag{2.39}$$

In case of four robots as shown in Figure 2.17b, the formation Jacobian $\mathbf{J_f}$ can be

derived by stacking Equation (2.39):

$$\dot{\mathbf{d}}_f = \mathbf{J_f v} \tag{2.40}$$

where:

$$\dot{\mathbf{d}}_\mathbf{f} = \begin{pmatrix} \dot{d}_{12} \\ \dot{d}_{13} \\ \dot{d}_{14} \\ \dot{d}_{23} \\ \dot{d}_{24} \\ \dot{d}_{34} \end{pmatrix}, \quad \mathbf{J_f} = \begin{pmatrix} -\mathbf{n}_{12} & \mathbf{n}_{12} & \mathbf{0}_{1\times2} & \mathbf{0}_{1\times2} \\ -\mathbf{n}_{13} & \mathbf{0}_{1\times2} & \mathbf{n}_{13} & \mathbf{0}_{1\times2} \\ -\mathbf{n}_{14} & \mathbf{0}_{1\times2} & \mathbf{0}_{1\times2} & \mathbf{n}_{14} \\ \mathbf{0}_{1\times2} & -\mathbf{n}_{23} & \mathbf{n}_{23} & \mathbf{0}_{1\times2} \\ \mathbf{0}_{1\times2} & -\mathbf{n}_{24} & \mathbf{0}_{1\times2} & \mathbf{n}_{24} \\ \mathbf{0}_{1\times2} & \mathbf{0}_{1\times2} & -\mathbf{n}_{34} & \mathbf{n}_{34} \end{pmatrix}_{6\times8}$$

$$\mathbf{n}_{ij} = \frac{(\mathbf{p_j} - \mathbf{p_i})^T}{d_{ij}}$$

From Equation (2.34), the formation task is, thus, defined as follows:

$$\mathbf{J_f v} = -\lambda_f \left(\mathbf{d_f} - \mathbf{d_f^*}\right) \tag{2.41}$$

where $\mathbf{d}$ and $\mathbf{d_f^*}$ are vectors of current and desired inter-distances, respectively. λ_f is the formation control gain.

Cooperative navigation

This task focuses on driving the formation's centroid toward a desired point $\mathbf{p_d} = (x_d, y_d)^T$ and orientation θ_d. The pose (position and orientation) of the centroid is used as the control feature. Figure 2.18 shows the task of team navigation, in which the centroid $\mathbf{p_c} = (x_c, y_c)^T$ has to move toward a target point $\mathbf{p_d}$ as well as to orient its heading to any desired angle, in that case, θ_d.

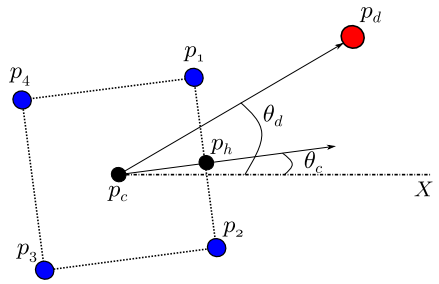


Figure 2.18 – Navigation scheme, the blue nodes represent robots whereas the red node indicates a target point.

The position of formation's centroid is defined as following, with n is the number of robots in the group:

$$x_c = \frac{1}{n}\sum_{i=1}^{n} x_i \quad \text{and} \quad y_c = \frac{1}{n}\sum_{i=1}^{n} y_i \tag{2.42}$$

The point $\mathbf{p_h} = (x_h, y_h)^T$ can be defined anywhere, relatively to each robot in the group. This point can be used to determine the heading of formation, and it can be expressed in relation to robots' positions as follows:

$$x_h = \sum_{i=1}^{n} a_i x_i \quad \text{and} \quad y_h = \sum_{i=1}^{n} a_i y_i \tag{2.43}$$

where a_i is the relation coefficient to i^{th} robot, and $\sum_{i=1}^{n} a_i = 1$. For instance, $a_1 = a_2 = 0.5$ and $a_3 = a_4 = 0$ in the configuration shown in Figure 2.18.

The group's heading can be found using $\mathbf{p_c}$ and $\mathbf{p_h}$:

$$\theta_c = \arctan\left(\frac{y_h - y_c}{x_h - x_c}\right) = \arctan\frac{\sum_{i=1}^{n} y_i\left(a_i - \frac{1}{n}\right)}{\sum_{i=1}^{n} x_i\left(a_i - \frac{1}{n}\right)} \tag{2.44}$$

By differentiating Equation (2.42) and (2.44) with respect to time, the Jacobian of navigation task $\mathbf{J_n}$ can be determined as follows:

$$\begin{pmatrix} \dot{x}_c \\ \dot{y}_c \\ \dot{\theta}_c \end{pmatrix} = \begin{pmatrix} 1/n & 0 & \cdots & 1/n & 0 \\ 0 & 1/n & \cdots & 0 & 1/n \\ -m_{y_1} & m_{x_1} & \cdots & -m_{y_n} & m_{x_n} \end{pmatrix} \begin{pmatrix} \dot{\mathbf{p_1}} \\ \vdots \\ \dot{\mathbf{p_n}} \end{pmatrix} \tag{2.45}$$

where:

$$m_{x_j} = \frac{(a_j - 1/n)\sum_{i=1}^{n} x_i\left(a_i - \frac{1}{n}\right)}{\left[\sum_{i=1}^{n} x_i\left(a_i - \frac{1}{n}\right)\right]^2 + \left[\sum_{i=1}^{n} y_i\left(a_i - \frac{1}{n}\right)\right]^2}$$

$$m_{y_j} = \frac{(a_j - 1/n)\sum_{i=1}^{n} y_i\left(a_i - \frac{1}{n}\right)}{\left[\sum_{i=1}^{n} x_i\left(a_i - \frac{1}{n}\right)\right]^2 + \left[\sum_{i=1}^{n} y_i\left(a_i - \frac{1}{n}\right)\right]^2}$$

$$j = 1, 2, ..., n$$

Applying to four robots configuration, we get:

$$\begin{pmatrix} \dot{x}_c & \dot{y}_c & \dot{\theta}_c \end{pmatrix}^T = \mathbf{J_n V} \tag{2.46}$$

where $\mathbf{J_n} = \begin{pmatrix} 1/4 & 0 & \cdots & 1/4 & 0 \\ 0 & 1/4 & \cdots & 0 & 1/4 \\ -m_{y_1} & m_{x_1} & \cdots & -m_{y_4} & m_{x_4} \end{pmatrix}$

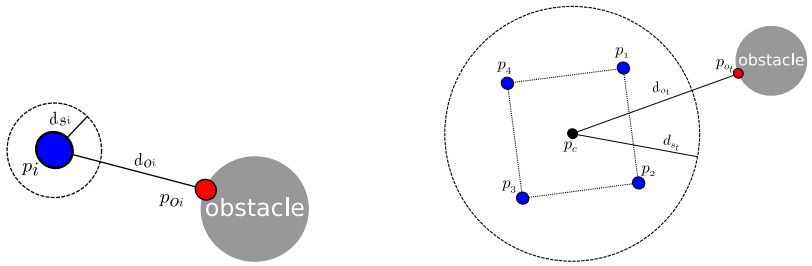
From Equation (2.34), the navigation task can be derived as follows:

$$\mathbf{J_n v} = - \begin{pmatrix} \lambda_p \left(\mathbf{p_c} - \mathbf{p_d} \right) \\ \lambda_\theta \left(\theta_c - \theta_d \right) \end{pmatrix} \tag{2.47}$$

where λ_p and λ_θ are the control gains for position and orientation, respectively.

Individual and team obstacle avoidance

Figure 2.19 illustrates the two obstacle scheme used to define their respective tasks. The red node represents a closest detected point of obstacle (Figure 2.19a) to robot, $\mathbf{p_{o_i}}$ and (Figure 2.19b) to group's centroid, $\mathbf{p_{o_t}}$. d_{o_i} and d_{o_t} are the distances from i^{th} robot and team's centroid to the obstacle, while d_{s_i} and d_{s_t} are the safety distances for individual and team obstacle avoidance, respectively.



(a) (b)

Figure 2.19 – Obstacle avoidance scheme: (a) individual and (b) team avoidance.

Two obstacle avoidance schemes are introduced in the overall control law depending on the situation (Figure 2.19). For instance, individual obstacle avoidance is useful when each robot tries to converge to a defined formation, while the team avoidance is more practical during navigation. They can be seen as sets of inequality constraints whose purpose is to make sure that the distances to the obstacle(s), which are the control features, can never be smaller than the safety distances [PAM20]. They can be found as follows:

$$d_{o_i} = \|\mathbf{p_{o_i}} - \mathbf{p_i}\| \quad \text{and} \quad d_{o_t} = \|\mathbf{p_{o_t}} - \mathbf{p_c}\| \tag{2.48}$$

Similar to the formation task, we differentiate Equation (2.48) with respect to time. For each obstacle, we get the following:

$$\begin{aligned} \dot{d}_{o_i} &= \frac{(\mathbf{p_i} - \mathbf{p_{o_i}})^T}{d_{o_i}} \dot{\mathbf{p}}_\mathbf{i} \\ \dot{d}_{o_t} &= \frac{(\mathbf{p_c} - \mathbf{p_{o_t}})^T}{d_{o_t}} \begin{pmatrix} 1/n & 0 & \cdots & 1/n & 0 \\ 0 & 1/n & \cdots & 0 & 1/n \end{pmatrix} \begin{pmatrix} \dot{\mathbf{p}}_\mathbf{1} \\ \vdots \\ \dot{\mathbf{p}}_\mathbf{n} \end{pmatrix} \end{aligned} \tag{2.49}$$

Consider for our four robots, the individual and team obstacle avoidance Jacobian, $\mathbf{J_{o_i}}$ and $\mathbf{J_{o_t}}$, respectively, are written as follows:

$$\begin{aligned} \mathbf{J_{o_i}} &= \frac{(\mathbf{p_i} - \mathbf{p_{o_i}})^T}{d_{o_i}} \\ \mathbf{J_{o_t}} &= \frac{(\mathbf{p_c} - \mathbf{p_{o_t}})^T}{4d_{o_t}} \begin{pmatrix} \mathbf{I}_2 & \mathbf{I}_2 & \mathbf{I}_2 & \mathbf{I}_2 \end{pmatrix}_{2\times 8} \end{aligned} \tag{2.50}$$

The inequality constraints of the obstacle avoidance tasks can be, therefore, expressed as follows:

$$\begin{aligned} \mathbf{J_{o_i}} \mathbf{v_i} &\geq -\lambda_{o_i} (\mathbf{d_{o_i}} - \mathbf{d_{s_i}}) \\ \mathbf{J_{o_t}} \mathbf{v_t} &\geq -\lambda_{o_t} (\mathbf{d_{o_t}} - \mathbf{d_{s_t}}) \end{aligned} \tag{2.51}$$

where λ_{o_i} and λ_{o_t} are the control gains for individual and team obstacle avoidance, respectively.

The distances can be in the form of vectors or scalars depending on the number of obstacles detected in the workspace, and they can also be null (thus no inequality constraint) if there is no presence of the obstacle.

Velocity limits

An upper and lower velocity limit is imposed on the robot's left and right wheel as inequality constraints. Consider b and r as, respectively, the distance between the two wheels and wheel's radius (as shown in Figure 2.1), the relation of linear and angular velocity (v_i, ω_i) to wheel's velocity $(\omega_{l,i}, \omega_{r,i})$ of i^{th} robot can be expressed as follows:

$$\begin{aligned} \omega_{l,i} &= \frac{1}{r} (v_i - 0.5b\omega_i) \\ \omega_{r,i} &= \frac{1}{r} (v_i + 0.5b\omega_i) \end{aligned} \tag{2.52}$$

Therefore, the inequality constraint of the velocity limit for four robots can be written as following:

$$\mathbf{J_{lim}}\mathbf{u} \leq \boldsymbol{\omega}_{lim} \tag{2.53}$$

where:

$\mathbf{J_{lim}}$ is a 16×8 block diagonal matrix of $\mathbf{G}$,

$$\mathbf{G} = \frac{1}{r}\begin{pmatrix} 1 & -0.5b \\ 1 & 0.5b \\ -1 & 0.5b \\ -1 & -0.5b \end{pmatrix},$$

$$\boldsymbol{\omega}_{lim} = \begin{pmatrix} \mathbf{H} \\ \mathbf{H} \\ \mathbf{H} \\ \mathbf{H} \end{pmatrix}, \mathbf{H} = \begin{pmatrix} \omega_{l,max} \\ \omega_{r,max} \\ -\omega_{l,min} \\ -\omega_{r,min} \end{pmatrix}$$

2.3.4 Simulation and experimental results

In order to validate the proposed algorithm, the control scheme is decomposed into two behavior states:

— Stage 1 (S1): Initial formation ensures that the desired shape is reached before mounting the payload on top. At this stage, only one level of optimization is needed. Hence, a formation task that is subjected to individual obstacle avoidance and velocity limits constraints, which leads to the initial formation.

— Stage 2 (S2): Cooperative navigation is responsible for driving the centroid to a target point, taking into account preservation of formation, team obstacle avoidance, and velocity limit. Therefore, two hierarchies are proposed. A higher level for solving the formation task, and a second priority for the navigation task. From Equation (2.36), the HQP problem can be written as follows:

$$\begin{aligned} \text{Level 1}: \quad & \mathbf{u_1} = \arg\min_{\mathbf{x}} && \|\mathbf{A_1 x} - \mathbf{b_1}\|^2 \\ & \text{s.t.} && \mathbf{C_1 x} \leq \mathbf{d_1} \\ \text{Level 2}: \quad & \mathbf{u} = \arg\min_{\mathbf{x}} && \|\mathbf{A_2 x} - \mathbf{b_2}\|^2 \\ & \text{s.t.} && \mathbf{A_1 x} = \mathbf{A_1 u_1} \\ & && \mathbf{C_2 x} \leq \mathbf{d_2} \end{aligned} \tag{2.54}$$

Table 2.2 summarizes the overall control law of the system. The control input is always a solution from the last hierarchy, which is solved using the pseudo-inverse approach in order to avoid having multiple sets of solutions. The hierarchical priority order of the tasks is velocity limits $\succ$ obstacle avoidance $\succ$ formation $\succ$ navigation task.

Table 2.2 – Hierarchy levels of tasks for the two behavior states

	S1: $\mathbf{u} \equiv \mathbf{u_f}$	S2: $\mathbf{u} \equiv \mathbf{u_n}$
Level 1	$\min_{\mathbf{u_f}} \left\|\mathbf{J_f J_x u_f} + \lambda_f \left(\mathbf{d_f} - \mathbf{d_f^*}\right)\right\|^2$ s.t. $\mathbf{J_{o_i} J_x u_i} \geq -\lambda_{o_i} \left(\mathbf{d_{o_i}} - \mathbf{d_{s_i}}\right)$ $\mathbf{J_{lim} u_f} \leq \boldsymbol{\omega}_{lim}$	$\min_{\mathbf{u_f}} \left\|\mathbf{J_f J_x u_f} + \lambda_f \left(\mathbf{d_f} - \mathbf{d_f^*}\right)\right\|^2$ s.t. $\mathbf{J_{lim} u_f} \leq \boldsymbol{\omega}_{lim}$
Level 2	none	$\min_{\mathbf{u_n}} \left\|\mathbf{J_n J_x u_n} + \begin{pmatrix} \lambda_p \left(\mathbf{p_c} - \mathbf{p_d}\right) \\ \lambda_\theta \left(\theta_c - \theta_d\right) \end{pmatrix}\right\|^2$ s.t. $\mathbf{J_f J_x u_n} = \mathbf{J_f J_x u_f}$ $\mathbf{J_{o_t} J_x u_n} \geq -\lambda_{o_t} \left(\mathbf{d_{o_t}} - \mathbf{d_{s_t}}\right)$ $\mathbf{J_{lim} u_n} \leq \boldsymbol{\omega}_{lim}$

The program always starts from S1 behavior until inter-distance errors between robots fall below a certain threshold before switching to S2. In addition, if there is any sudden increase of the inter-distance errors above a threshold, the controller will transition back to S1 to ensure the formation before moving again.

Similar to the experimental setup in the previous section, ROS and Gazebo are used to simulate the robots and environment for the simulation. For the real experiment, all robots in the team send their localized position to a PC for performing the optimization. The resultant velocities are then broadcast back to each corresponding robot for motion control.

Simulation

Two simulations are done with different numbers of robots in the formation. In both cases, the controller respects the hierarchy levels described previously. In addition, the graphs for the two simulations are fully-connected and rigid. Hence, the number of features

for the formation task is six and nine for simulations I and II, respectively. The desired inter-distance for the side neighbors is 2.0m.

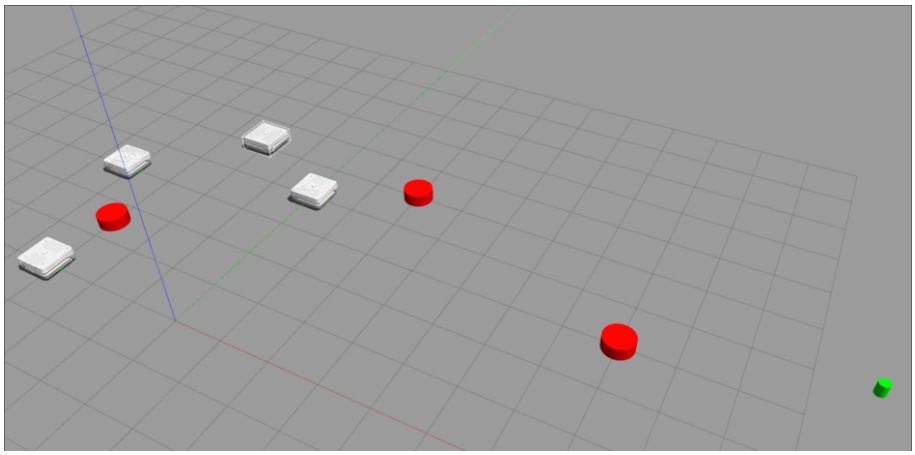
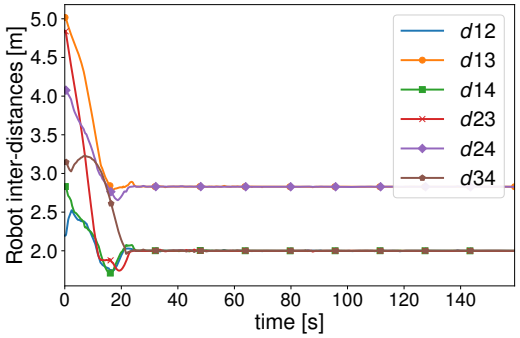
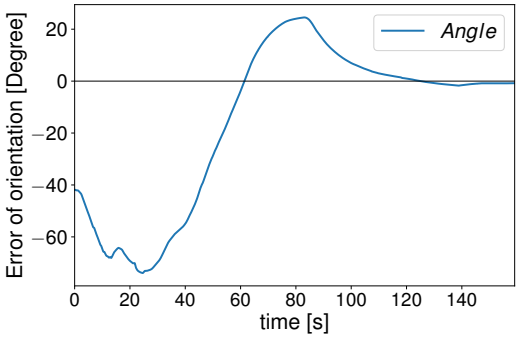


Figure 2.20 – Environment of simulation I.

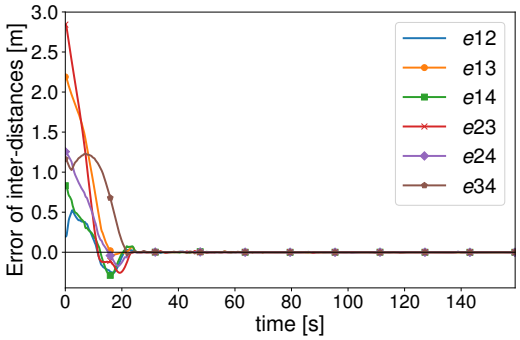
Simulation I: A group of four robots starts the formation at a less-than-ideal shape. Alongside the robots, there are three obstacles in the environment as shown in Figure 2.20. Figure 2.21a shows the convergence of inter-distances to the desired values at around $t = 10$s; this marks the end of stage S1 and begins the cooperative navigation phase. Figure 2.21c illustrates the error of inter-distances with respect to the desired values; these inter-distances are well maintained with a very small deviation from the set desired values ($\pm$7.5mm) as shown in Figure 2.21d. During this time, the formation heading angle also converges a set value of 90^o as shown in Figure 2.21b, where the error converges nicely to zero. Last but not least, the overall trajectories are demonstrated in Figure 2.21e. The blue square is the formation shape when stage S1 is completed.



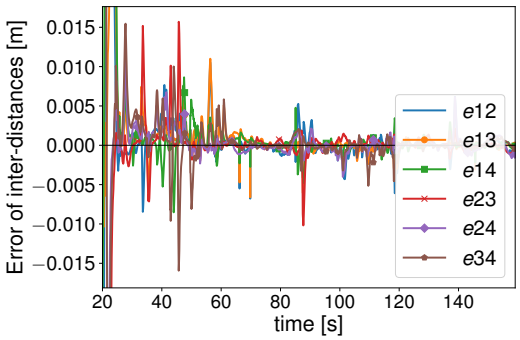
(a) Inter-distances between robots.



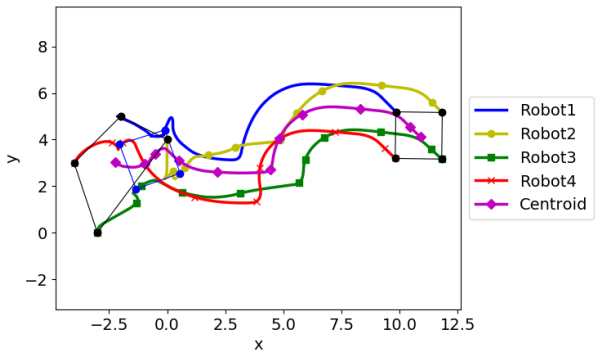
(b) Heading error.



(c) Errors of inter-distances.



(d) Magnified inter-distances errors.



(e) Robots' trajectories.

Figure 2.21 – Plots of simulation I.

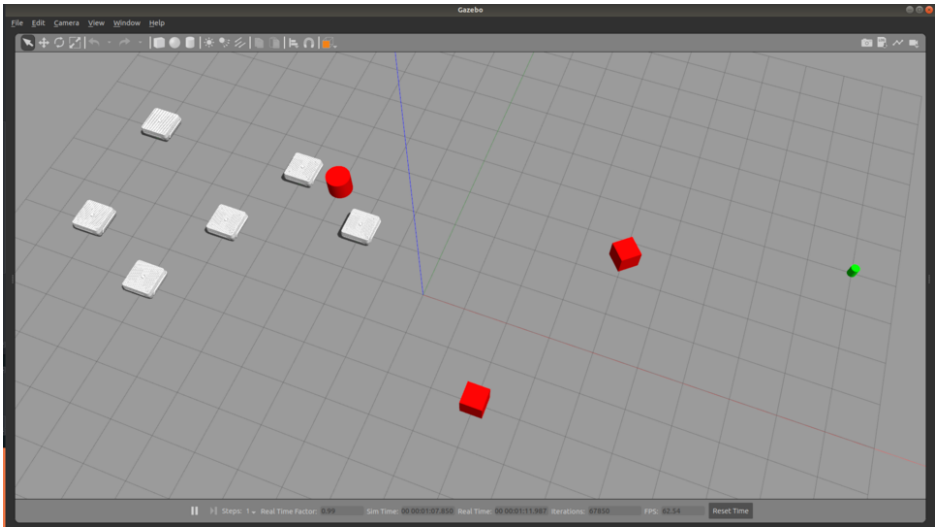
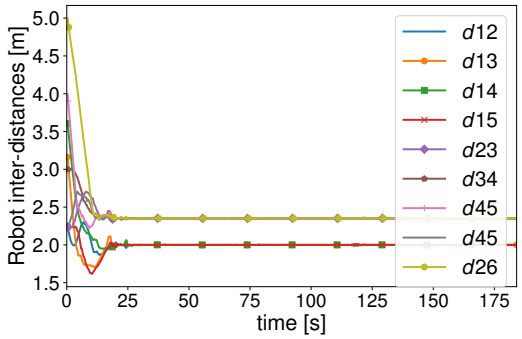


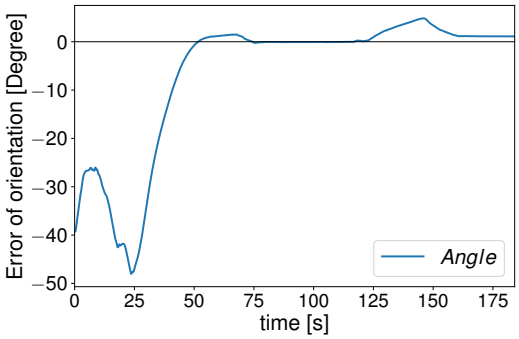
Figure 2.22 – Environment of simulation II.

Simulation II: As shown in six robots and three obstacles are utilized in this simulation. Figure 2.22 shows the simulation environment, in which there are three obstacles: a round, a square, and a rotated square.

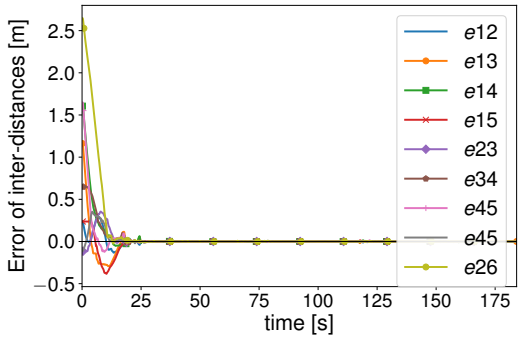
From Figure 2.23a, we can see that the inter-distance errors converge to form the formation (S1) at around $t = 25$s. After that, the robots start to move to a goal point with the desired formation heading of -90^o. The team's orientation error is well-converged with an error of less than 5^o as illustrated in Figure 2.23b. The shape is preserved throughout the navigation as shown in Figure 2.23c. The maximum inter-distance error is around ± 0.018m even though they have to navigate through obstacles as demonstrated in Figure 2.23d. Since the readability of the trajectories plot is increased as the number of robots increases, snapshots of the Gazebo are instead provided to aid in visualizing the formation shape; Figure 2.24a and Figure 2.24b show the shape at the end of stage S1 and S2, respectively.



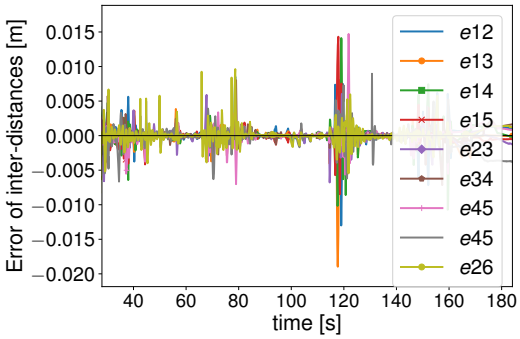
(a) Inter-distances between robots.



(b) Heading error.

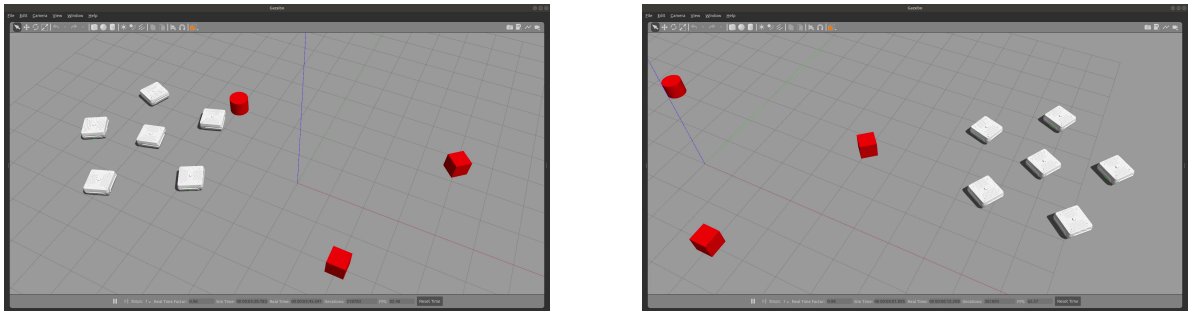


(c) Errors of inter-distances.



(d) Magnified inter-distances errors.

Figure 2.23 – Plots of simulation II.



(a) Formation shape at $t = 25$s.

(b) End of the formation.

Figure 2.24 – Formation shape of simulation II.

Experiment

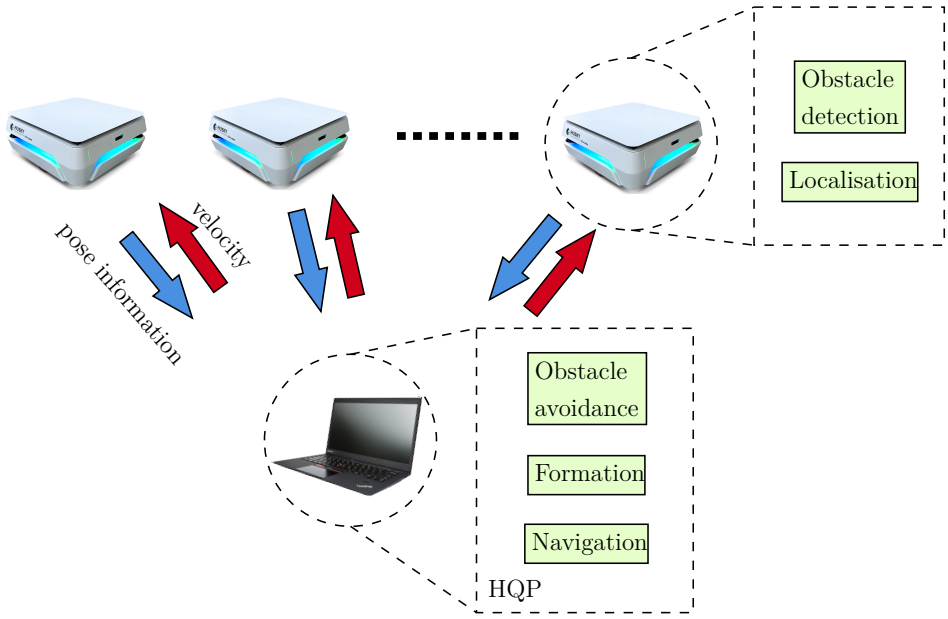


Figure 2.25 – Experimental setup of the HQP formation.

In this experiment, we separate the stages S1 and S2 into two experiments because the group has to first get into a good formation shape, then the payload is manually attached before starting the navigation. In both cases, a square shape formation of sides of 2m is desired. ROS is used as the middleware. Each robot has its own independent *ROS_MASTER*. There is no external sensor to locate each robot accurately. The self-localization in a map is done using an onboard lidar sensor. In contrast to the distributed local controller of consensus, the control velocities for each robot are computed by a

centralized optimization solver, an architecture of which is shown in Figure 2.25.

S1: Figure 2.26 shows the actual initial poses of the robots, which is not yet a square shape. In addition, there exists an obstacle in order to trigger the individual obstacle avoidance inequality constraint.

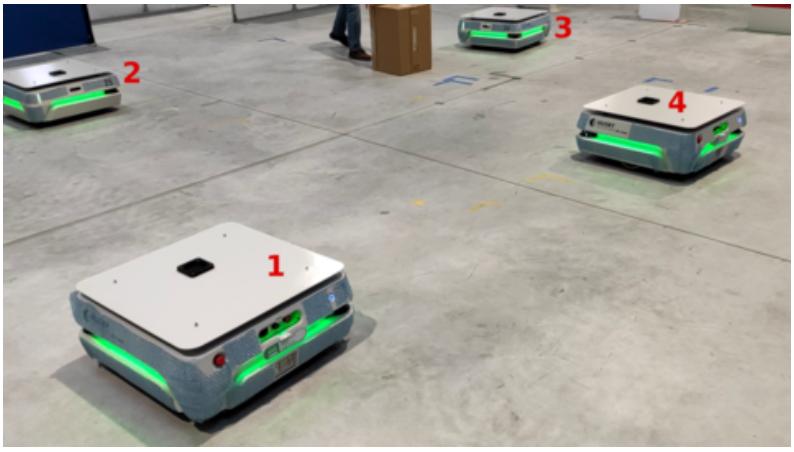
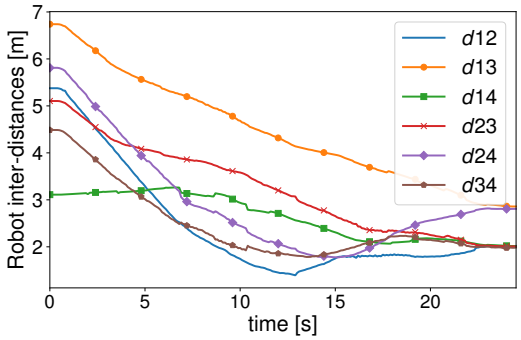
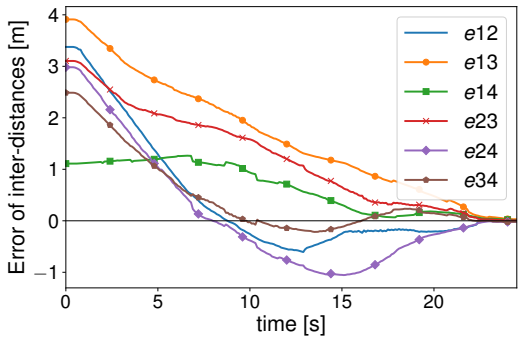


Figure 2.26 – Initial poses at S1.

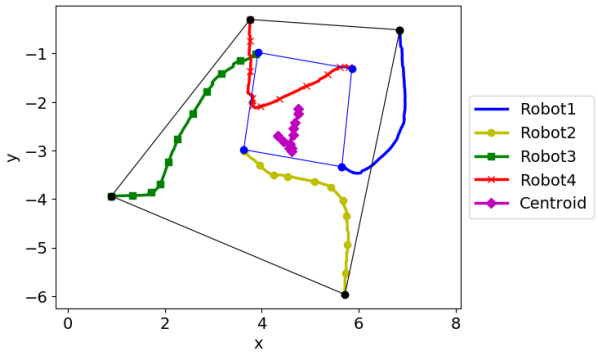
From Figure 2.27a, we can see that the overall formation converges to the square shape as the inter-distances between side neighbors converges to the desired value of 2m. It is also solidified by the graph of the errors shown in Figure 2.27b; they converge nicely to zero. The overall trajectories of the robots are illustrated in Figure 2.27c. At the end of S1, the desired formation shape is achieved (blue square in Figure 2.27c), and ready to attach the load to start S2.



(a) Inter-distances between robots.



(b) Errors of inter-distances.



(c) Robots' trajectories.

Figure 2.27 – Plots of S1 experiment.

S2: Figure 2.28 shows the navigation scenario of the experiment, in which a plate with a payload of 20kg is added to the system. The team has to reach a target point and come back while trying not to collide with the two un-mapped obstacles.

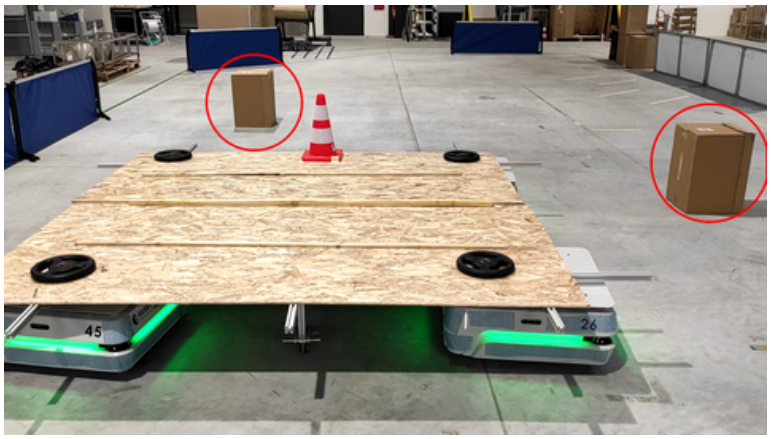
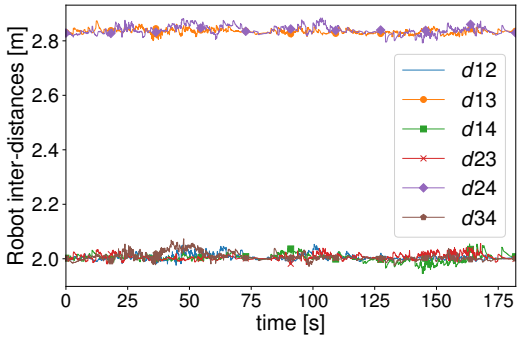


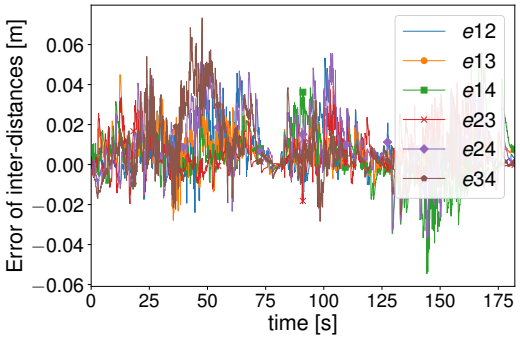
Figure 2.28 – Navigation scenario with two obstacles.

The formation shape is kept during navigation, as demonstrated by the plot of the inter-distance errors in Figure 2.29a. Even though the self-localization, as well as the velocity tracking low-level control of each robot, are not accurately perfect, the maximum error for the inter-distances is just about$\pm$0.06m, evidenced in Figure 2.29b. The fleet starts at (2.5m, $-$2.0m, 0^o), and moves to a goal pose of (12m, $-$4.0m, 90^o). Once it reaches the desired pose at around $t = 80$s, it has to return to its initial pose. The trajectories of each robot as well as the formation's center can be seen in Figure 2.29d; the blue square indicates the formation shape at the start and end of the navigation,

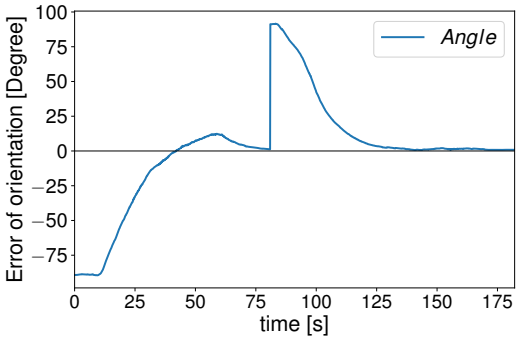
whereas the black square represents the shape midway when the fleet reaches the goal before returning. The team's desired heading angle is also reached for both journeys as the error converges to zero as demonstrated in Figure 2.29c.



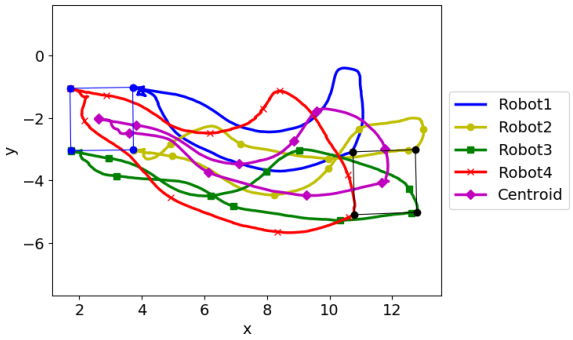
(a) Inter-distances between robots.



(b) Errors of inter-distances.



(c) Heading error.



(d) Robots' trajectories.

Figure 2.29 – Plots of S2 experiment.

In order to show the effectiveness of the proposed approach, we also perform more experiments using a reduced number of inter-distance features. As the goal is to see its effectiveness for different formation configurations, we do not need to attach any load to the fleet. Therefore, we can combine the two stages S1 and S2 in one run for each test. As discussed in the rigidity part of Section 2.2.2, one can define a minimally rigid graph and the formation shall still be fixed. To validate this, we try the proposed HQP controller with a team of three, four, and five robots with a reduced number of edges $(2n-3)$. All following experiments start with initial poses that require the fleet to do formation phase S1 first before navigating to a goal point while avoiding one un-mapped obstacle.

Four robots: Figure 2.30 shows the topology of the experimental setup for four robots, which we use the concept of minimally rigid graph resulting in only five (using Equation (2.10)) inter-distances required for the formation, instead of six (using Equation (2.8)).

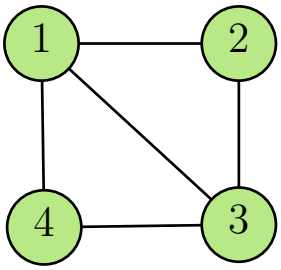
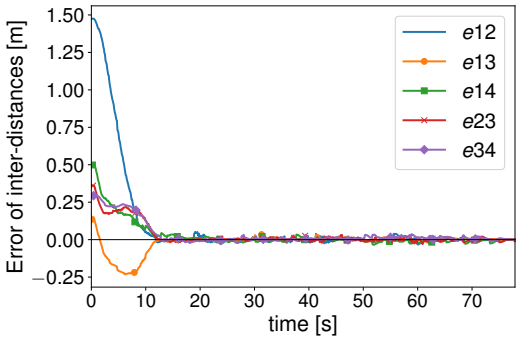
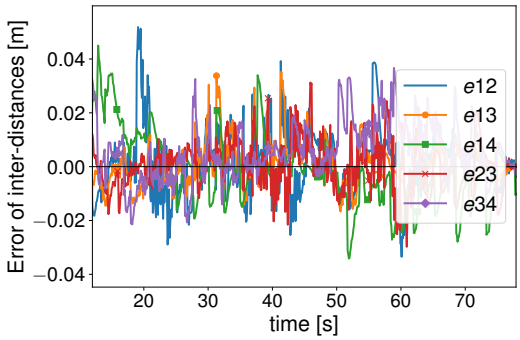


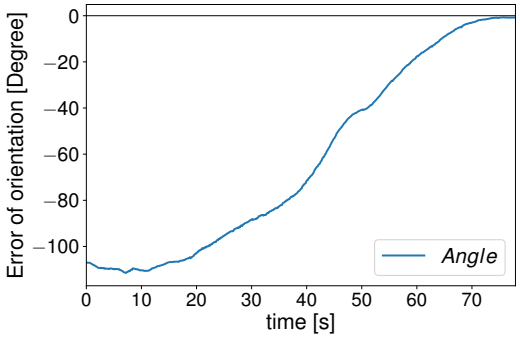
Figure 2.30 – Reduced inter-distances edges of four-robots team.



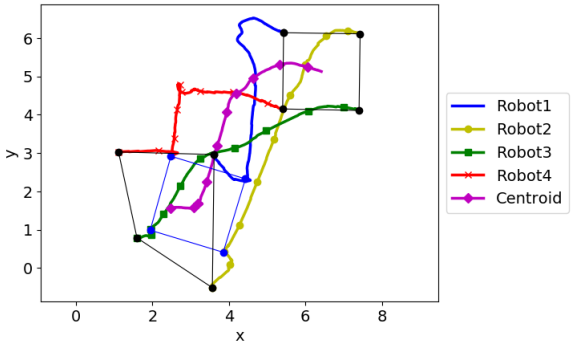
(a) Errors of inter-distances.



(b) Magnified inter-distances errors.



(c) Heading error.



(d) Robots' trajectories.

Figure 2.31 – Experiment of four robots with reduced number of inter-distance features.

Figure 2.31 shows the testing result of a square formation. The shape is reached at around $t = 12$s as shown in Figure 2.31d. Then, the fleet starts to move to the goal point without breaking up as the inter-distances errors remain below ± 0.05m, illustrated in Figure 2.31a and Figure 2.31b. The team's heading also reaches the desired value since its error converges nicely to zero as demonstrated in Figure 2.31c.

By simply changing the values of desired inter-distances, we can define any formation shape we want. For instance, Figure 2.32 shows a shape that is similar to an antenna can be achieved using the same graph topology as the previous case, but with a different set of values for the inter-distances.

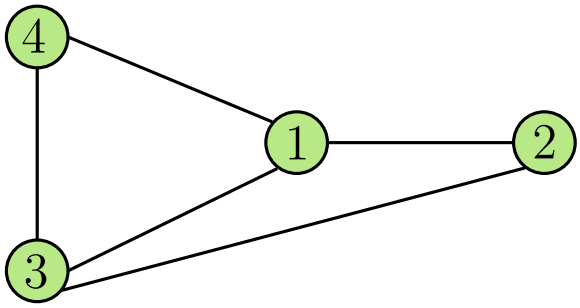
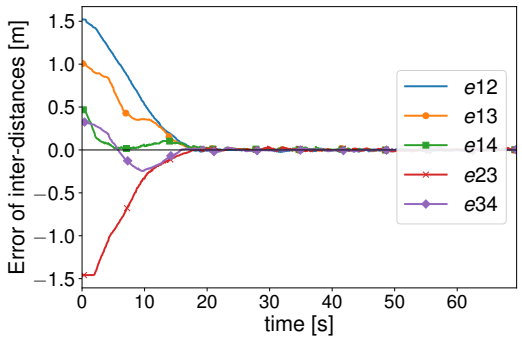
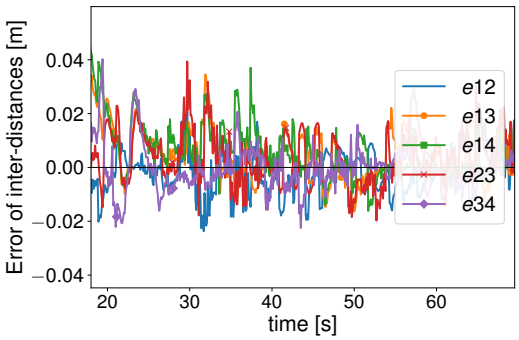


Figure 2.32 – Antenna shape of the formation.

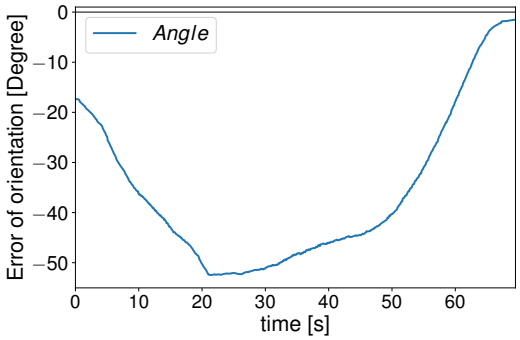
The experimental result is shown in Figure 2.33. From a bad initial formation shape, the team of robots reaches the desired shape at around $t = 15$s. Then they start to navigate to the goal point without any collision with the obstacle. The errors of inter-distances remain well under ± 0.04m.



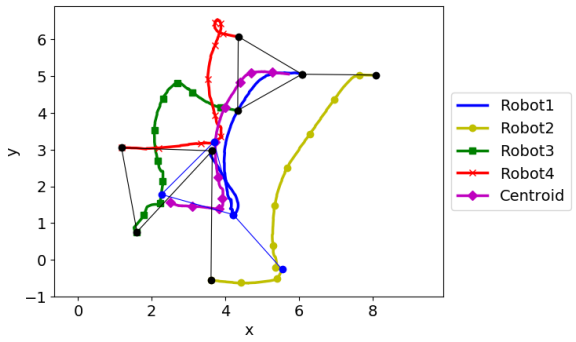
(a) Errors of inter-distances.



(b) Magnified inter-distances errors.



(c) Heading error.



(d) Robots' trajectories.

Figure 2.33 – Experiment of four robots with reduced number of inter-distance features and modified desired inter-distances.

Five robots: Figure 2.34 shows five-robots formation configuration. By following the minimally rigid graph rule, the required number of connected robots are seven (from Equation (2.10)), instead of ten (from Equation (2.8)).

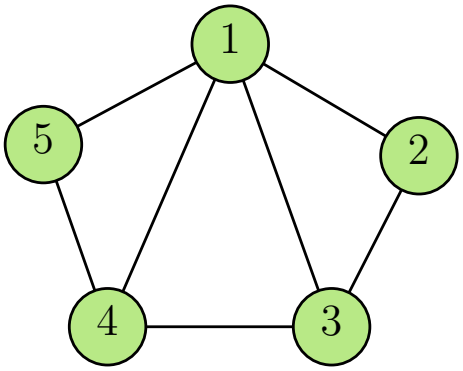
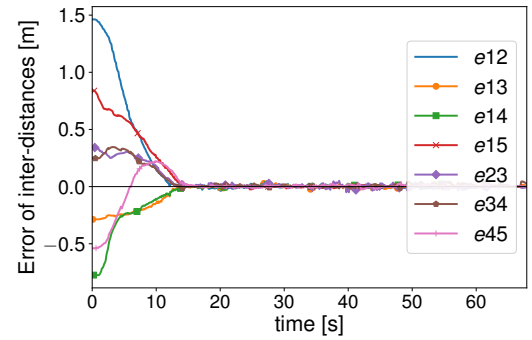
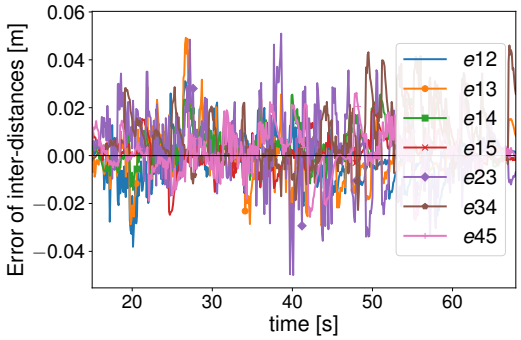


Figure 2.34 – Reduced inter-distances edges of five-robots team.

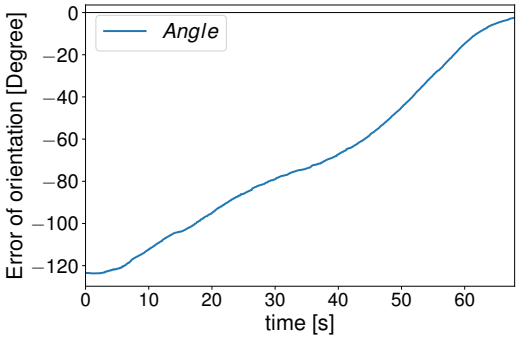
Figure 2.35 shows the testing result of a pentagon formation. Once again, the desired shape is achieved at around $t = 15$s as shown in Figure 2.31d. While moving to the goal point, the fleet manages to keep the inter-distances close to the desired values as shown by the minimal value of errors in Figure 2.35a and Figure 2.35b. Figure 2.35c shows that the team's heading can reach the desired value with an error of less than 2^o. The trajectories of each robot and the fleet are demonstrated in Figure 2.35d.



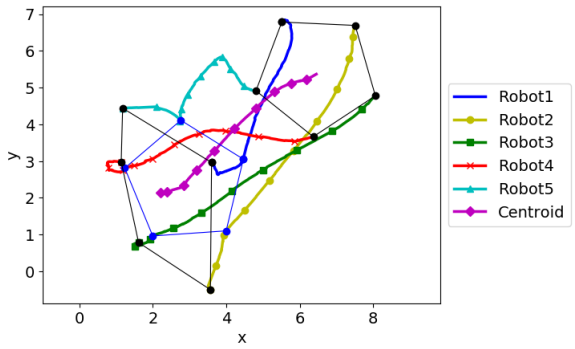
(a) Errors of inter-distances.



(b) Magnified inter-distances errors.



(c) Heading error.



(d) Robots' trajectories.

Figure 2.35 – Experiment of five robots with reduced number of inter-distance features.

Three robots: Figure 2.37 shows the inter-distance configuration in the case of three robots. In this case, the required number of connected edges/robots can not be reduced further as apply the Equation (2.8) and (2.10) gives the same result of three.

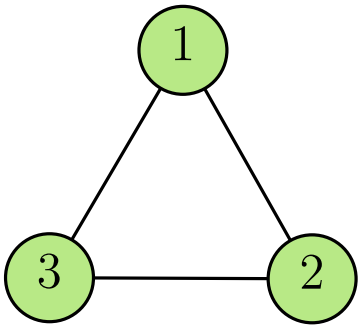
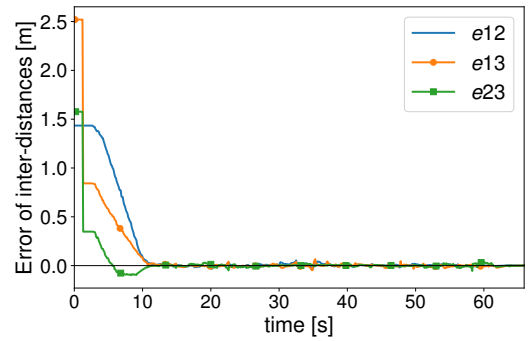
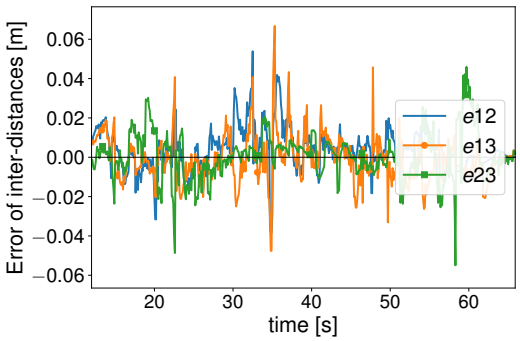


Figure 2.36 – Three-robots configuration.

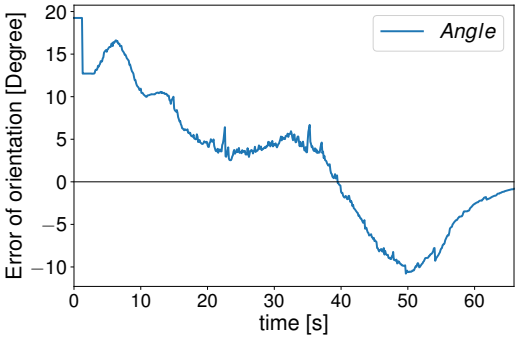
Figure 2.37 illustrates the testing result of a triangle formation. At $t = 12$s, the formation shape in stage S1 is achieved. Then S2 starts to navigate the fleet to the goal point, during which the shape remains fixed as shown in Figure 2.37d. Similar to previous experiments, the inter-distances errors are largely below ± 0.06m as demonstrated in Figure 2.31a and Figure 2.31b. The team's heading is eventually reached the desired value as shown in Figure 2.31c.



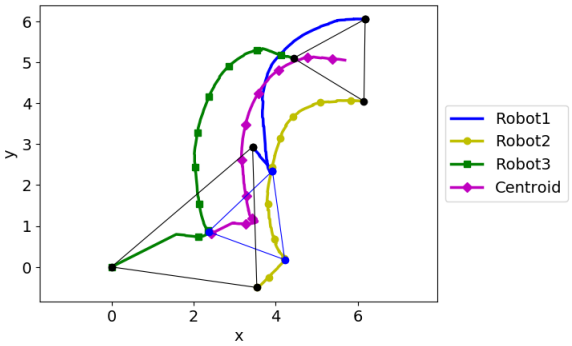
(a) Errors of inter-distances.



(b) Magnified inter-distances errors.



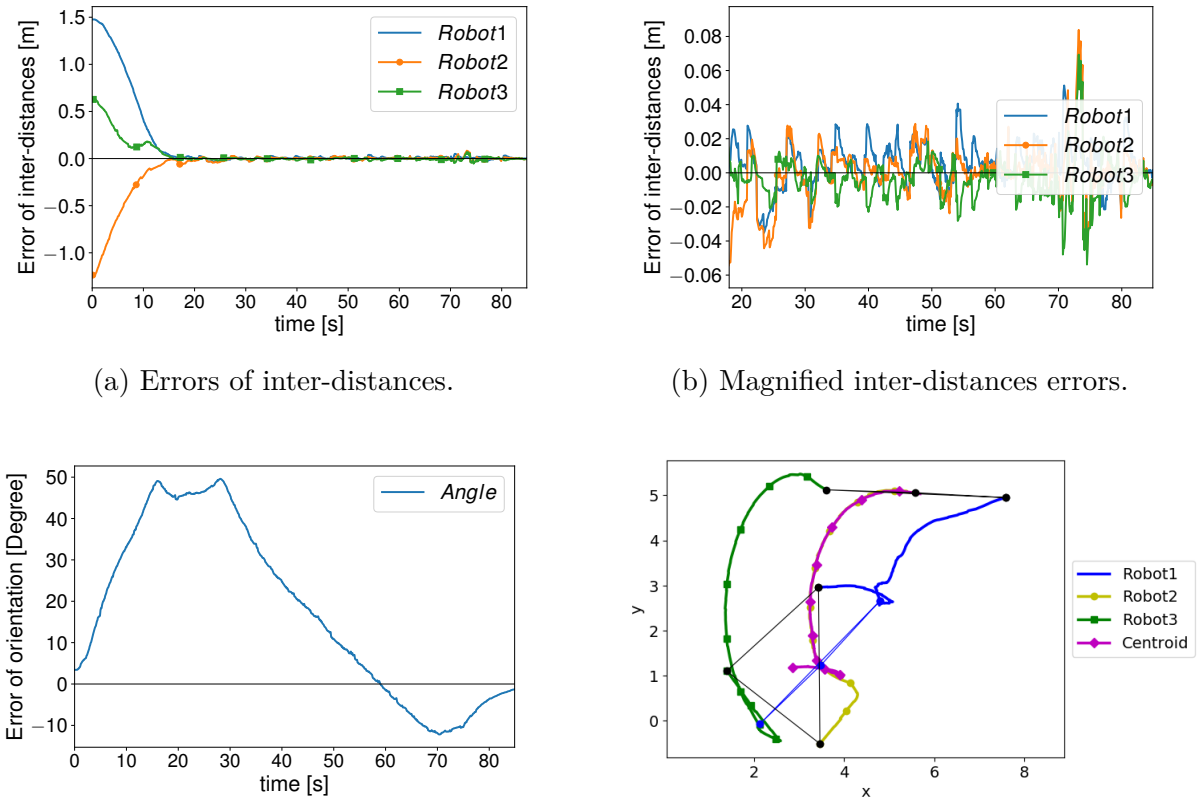
(c) Heading error.



(d) Robots' trajectories.

Figure 2.37 – Experiment of a triangle formation with three robots.

Last but not least, a line formation experiment is done to show that the proposed controller is robust in terms of the different configurations of the formation shape; the result of which can be seen in Figure 2.38. Once again, after reaching the initial formation stage, the fleet can navigate to the goal point while avoiding any collision and keeping the formation shape. The desired orientation converges to zero, while the errors of inter-distances are within ± 0.08m as shown in Figure 2.38c and Figure 2.38b, respectively.



(a) Errors of inter-distances.

(b) Magnified inter-distances errors.

(c) Heading error.

(d) Robots' trajectories.

Figure 2.38 – Experiment of a line formation with three robots.

2.3.5 Conclusion

In this section, we present a method for cooperative object transportation with obstacle avoidance based on an optimization approach. Different tasks were formulated to either equality or inequality constraints, which are solved using the hierarchical quadratic programming method. Simulations and actual experiments with different numbers of robots and formation shapes are conducted in order to validate the effectiveness of the proposed approach. Un-mapped obstacles in the environment can be detected and evaded during runtime reactively. Videos of some of the experiments can be found following these links [5].

5. https://youtu.be/LYJWcOHd1L8
https://youtu.be/E1OdhVi11DU

2.4 Conclusion on formation control

We have discussed the two proposed formation controllers in this chapter. The first approach is based on the consensus algorithm in order to reach an equilibrium in terms of inter-distances between robots in the team. Graph theory is presented as a basis for modeling the communication between neighbors, while the concept of static state feedback is used in order to deploy the controllers in real nonholonomic mobile robots. Simulations and experiments are shown to validate the proposed consensus formation controller. In both situations, the fleet can perform the formation to reach a specific shape and keep it during navigating and obstacle avoidance. Despite enjoying the benefit of being a distributed controller, however, this approach suffers a problem of not being able to clearly impose a strict priority for each task; this results in having a fleet behavior that can not ensure precisely desired inter-distances between neighbors.

On the other hand, task priority can be introduced and respected in the hierarchical quadratic programming approach, which corresponds to our second contribution of the chapter. Using the concept of task-based control, we can define the Jacobian matrix that relates the robots' velocities to any feature we want to control, the inter-distances in this case. Tasks defined for this approach are: 1) geometric shape formation, 2) cooperative navigation, 3) individual and team obstacle avoidance, and 4) velocity limits. We introduce two levels of the optimization problem, which can be solved using HQP. Simulations and actual experiments are presented to evaluate the controller performance. Especially for the real experimental part, we have shown that the proposed controller can be used with a different number of robots to achieve various formation shapes. By applying the rules of the minimally rigid graph, the complexity of the optimization problem is linearly proportional to the number of robots in the fleet as the number of connected neighbors required is $2n - 3$ compared to the full $n(n-1)/2$ number of edges.

As the formation control enables a group of robots to cooperatively carry a load, it can not decide which robot should be chosen for the formation team; this is why multi-robot task allocation is needed in order to adequately choose and team up robots in a group ready for the cooperative transporting task. This will be the main purpose of the next chapter.

Chapter 3

Task allocation

Contents

3.1 Introduction

Task allocation is a requirement for multi-robot systems working in dynamic environments. An efficient task allocation model allows the robots to adjust their workloads in response to other robots' actions or system's missions to increase overall system performance.

This chapter presents an allocation problem that deals with tasks that requires single or multiple robots to complete. The tasks can be allocated statically at a fixed interval or dynamically as soon as they arrive. Two existing task allocation techniques were compared in order to find the most suitable approach for our application. The first technique is based on the market-based approach, namely Contract Net Protocol (CNP). CNP is a distributed allocation approach that relies on each robot to bid in order to compete for

obtaining the task. Tabu Search (TS) is the second technique, which we look into. It is a centralized optimization-based approach.

The chapter starts with the problem statement, which can be found in Section 3.2. Section 3.3 describes the algorithms of task allocation techniques we use. Last but not least, simulation results of a various number of robots and tasks are detailed in Section 3.4.

3.2 Problem definition

Let us consider a team of n robots, $R = \{R_1, R_2, \cdots, R_n\}$, and a set of m tasks $T = \{T_1, T_2, \cdots, T_m\}$, the potential solution for allocation shall reflect an optimal list of allocations; an example of which can be as $[R_1\ T_1\ T_2\ R_2\ T_3\ T_4\ T_5\ \cdots]$. Multi-robot task allocation (MRTA) is an optimization problem aiming to solve this allocation by minimizing a cost function or maximizing a profit. Therefore, such a problem can be expressed mathematically as follows:

$$\begin{gathered} \min \sum_{i=1}^{n}\sum_{j=1}^{m} w_{ij} f_{ij} \quad \text{or} \quad \max \sum_{i=1}^{n}\sum_{j=1}^{m} w_{ij} f_{ij} \\ f_{ij} \in \{0,1\}, \quad \forall i \in [1,n],\ \forall j \in [1,m] \end{gathered} \tag{3.1}$$

where w_{ij} is the cost, which can be the distance to travel, time for completion, etc., or reward, depending on the optimization type, for assigning robot i to task j. f_{ij} is equal to 1 if the i^{th} robot is assigned to j^{th} task, and 0 otherwise.

The optimization problem shown in Equation (3.1) can be subjected to various constraints depending on the specific application of allocation; some of the most common ones are as follows:

$$\sum_{i=1}^{n} f_{ij} = L_j, \quad \forall j \in [1,m] \tag{3.2}$$

$$\sum_{j=1}^{m} f_{ij} \leq L_i, \quad \forall i \in [1,n] \tag{3.3}$$

where L_j represents the number of robots required for task j and L_i is the maximum number of assignments allowed to each robot. Equation (3.2) ensures that an appropriate amount of robots is allocated to a task; $L_j = 1$ in the case of single-robot task. Depending on nature of the application, L_i of Equation (3.3) can be equal to 1; it corresponds to the single-assignment problem [WVB19].

For our application, the task allocation is deployed for logistics in warehouse environments. The problem focuses on having a list of tasks executed in less time possible. In addition, it has to be able to deal with tasks that are added in real-time. We focus on an environment with up to 10 robots and 40 tasks. The robots are single-task robots, thus they can do only one task at a time, but they are able to have multiple tasks in the queue. We assume that each robot has the same skills and velocity. Tasks are pick-up and drop-off missions. The robots need to pick up an object, move to a new location, and drop them off. The tasks can also require several robots working together to complete. Therefore, they can be either single-robot or multi-robot tasks. We assume that the time for picking and dropping an object is negligible compared to the travel time. Each task has a priority level to indicate the urgency for completion.

The allocation of tasks can be done on two levels. Initially, there is a set of tasks to be assigned, hence static allocation. Then, tasks can be added online for allocation dynamically. This dynamic allocation assigns high-priority tasks to available robots as soon as the task is introduced; this is level one allocation. The remaining unassigned tasks are added to the level two allocation loop, namely static allocation, which runs at a fixed slower frequency

In order to evaluate the two approaches: CNP and TS, we focus on optimizing the overall time needed to perform the task. Therefore, the travel time of each robot for the tasks is considered as the cost objective.

3.3 Methodology

In this section, we discuss the formulation of the cost function that is used for allocating tasks, which is based on the estimated travel time of each robot. Then we discuss in detail the Contract Net Protocol and Tabu Search algorithm, as well as a simplified Local Search approach. The idea is to use CNP as the set of initial solutions for the TS, which could enable us to better optimize the solutions with less search iteration.

3.3.1 Objective function

The objective function is a minimization problem with a cost that is based on the traveling time of the robots. From Equation (3.1), the function can be expressed as follows:

$$\begin{aligned} &\min \sum_{i=1}^{n} \sum_{j=1}^{m} w_{ij} f_{ij} \\ &f_{ij} \in \{0,1\}, \quad \forall i \in [1,n],\ \forall j \in [1,m] \end{aligned} \tag{3.4}$$

where w_{ij} is the estimated time for robot i to complete task j. The traveling distance can also be chosen, but we have decided to work with the time because it is able to easily incorporate the waiting time of the robot when dealing with multi-robot tasks (tasks that require more than one robot to complete). If one robot arrives first, it has to wait for others in order to start performing the mission together. Therefore, the estimated time w_{ij} includes: the time to travel from the current position to the pick-up point, the time to travel from the pick-up point to the drop-off point, and the waiting time in case of the multi-robot tasks.

3.3.2 Contract Net Protocol approach

The first algorithm to be tested to solve this Multi-Robot Task Allocation (MRTA) problem is the Contract Net Protocol (CNP). It is an auction-based approach that consists of four stages [BCD19]:

1. Announcement: an auctioneer broadcasts the task to other robots.
2. Submission: each robot computes a cost for performing that task based on a defined cost function.
3. Selection: the auctioneer chooses a winner with the lowest bid.
4. Contract: information of who is the winner announced by the auctioneer.

Figure 3.1 presents the flowchart of the CNP algorithm. It runs each time an allocation of one task to the robots is needed. Thus, the input of this function is the task to allocate. This process is repeated as many times as the number of tasks to allocate. The bids are made by all the robots that are available for the tasks. A robot is set to be available if it has not started moving or has just finished a task. The bids of robots are added to a list of bids that are then sorted to select the lowest bid(s). There are as many bids selected as the number of robots required by the task (in the case of multi-robot tasks).

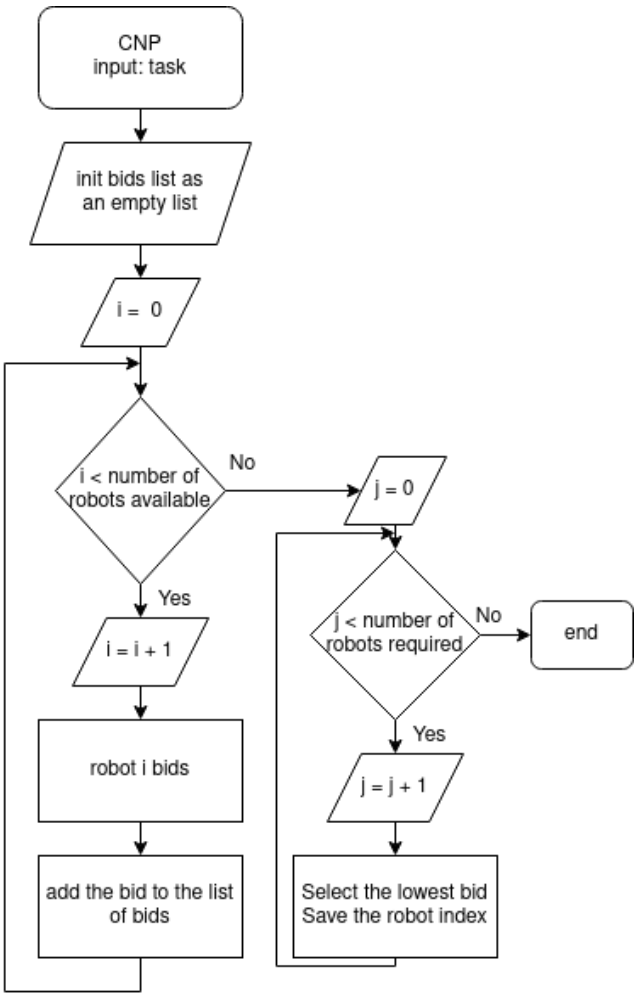


Figure 3.1 – Contract Net Protocol flowchart.

3.3.3 Tabu Search approach

The second algorithm that is tested is the Tabu Search algorithm. The flowchart of this algorithm is presented in Figure 3.3. This is a Local Search method [LRV14], which starts with a set of solution as initialization. Then, at each iteration of the algorithm, it looks for the neighborhood of this solution. A neighbor of a solution is the solution itself with two elements inverted. The process of finding the neighborhood is described in the flowchart of Figure 3.2. Once the neighborhood is found, the best neighbor is selected as being the new solution. The best neighbor is the one with the lowest cost, so the one that minimizes the most the total traveling time of the robots. Because this process can lead to cycling on a local minimum if the previous solution is the best neighbor of the new

one, the notion of Tabu is introduced. A Tabu is set to make it impossible to invert the two elements of the current solution that would lead to the past solution. This Tabu is kept for several iterations of the Tabu Search and is then removed to free some memory. By doing this, cycling on a local minimum is avoided. However, if this local minimum is the best solution found by the algorithm when it comes across it, it is set as being the global best solution. Each time a local optimum is found it is compared to the global best solution to know if it should be kept in memory or not.

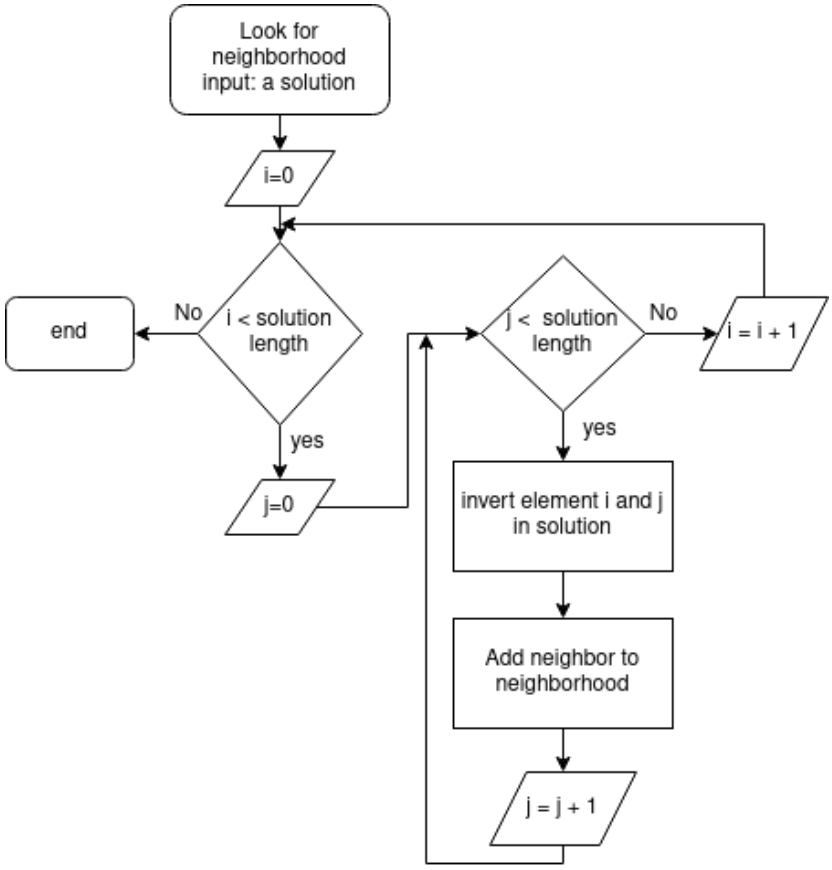


Figure 3.2 – Finding a neighborhood flowchart.

3.3.4 Simplified Local Search approach

The iterations of Tabu Search require a lot of computational resources and time. Indeed, finding all the neighbors of a solution requires m^2 operations, where m is the number of tasks. The initialization solution can be random or not. In order to improve

the efficiency of the TS algorithm, we start with a set of initial solutions obtained using CNP. Thus, the solution is close to optimal already.

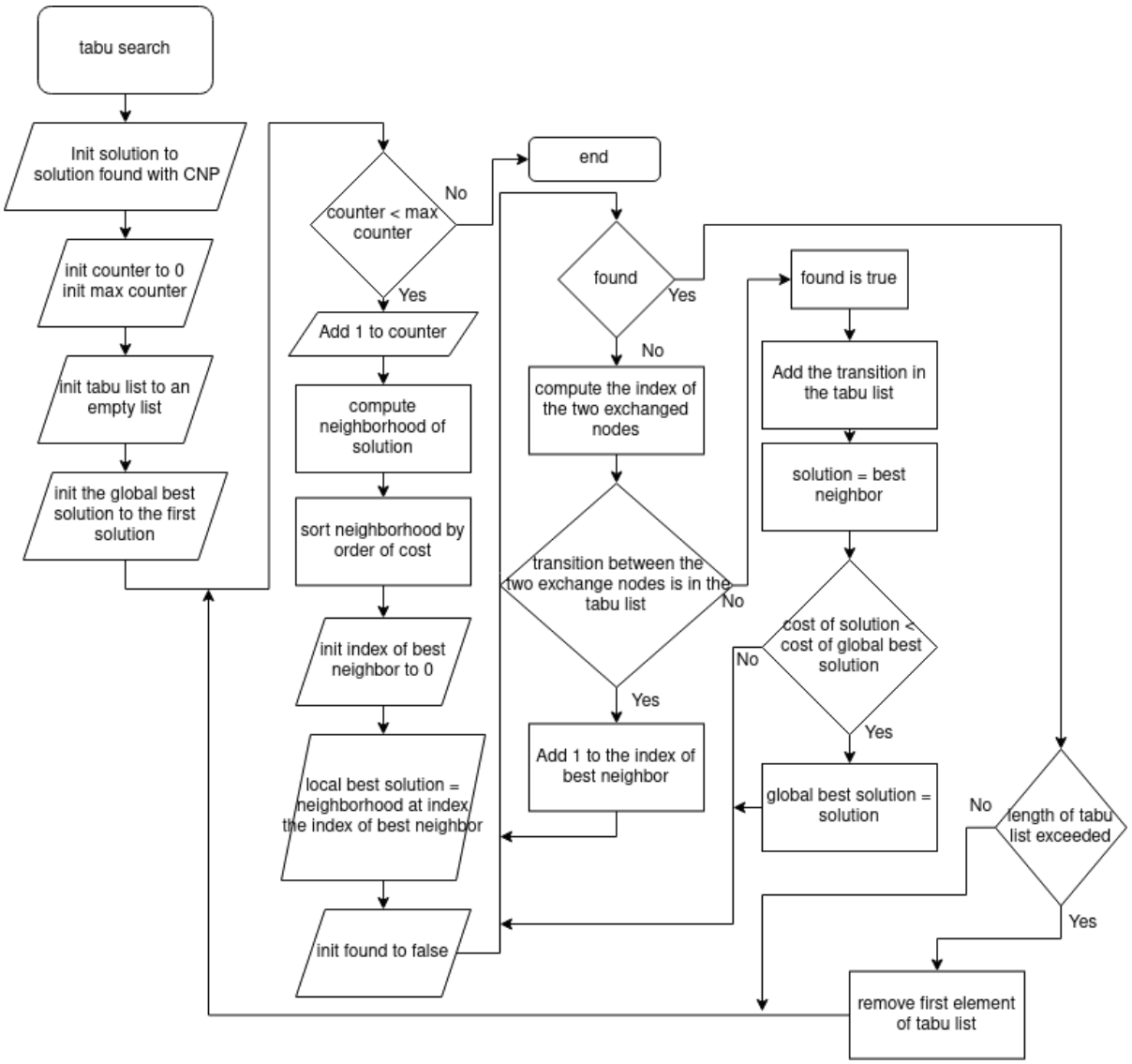


Figure 3.3 – Tabu Search flowchart.

A simplified version of the Tabu Search algorithm would be to use the fact that the initialization is done using the CNP solution to do a simple Local Search; it corresponds to one iteration of the previous Tabu Search algorithm without taking into account the notion of Tabu. This simplified Local search algorithm is presented in the flowchart of Figure 3.4.

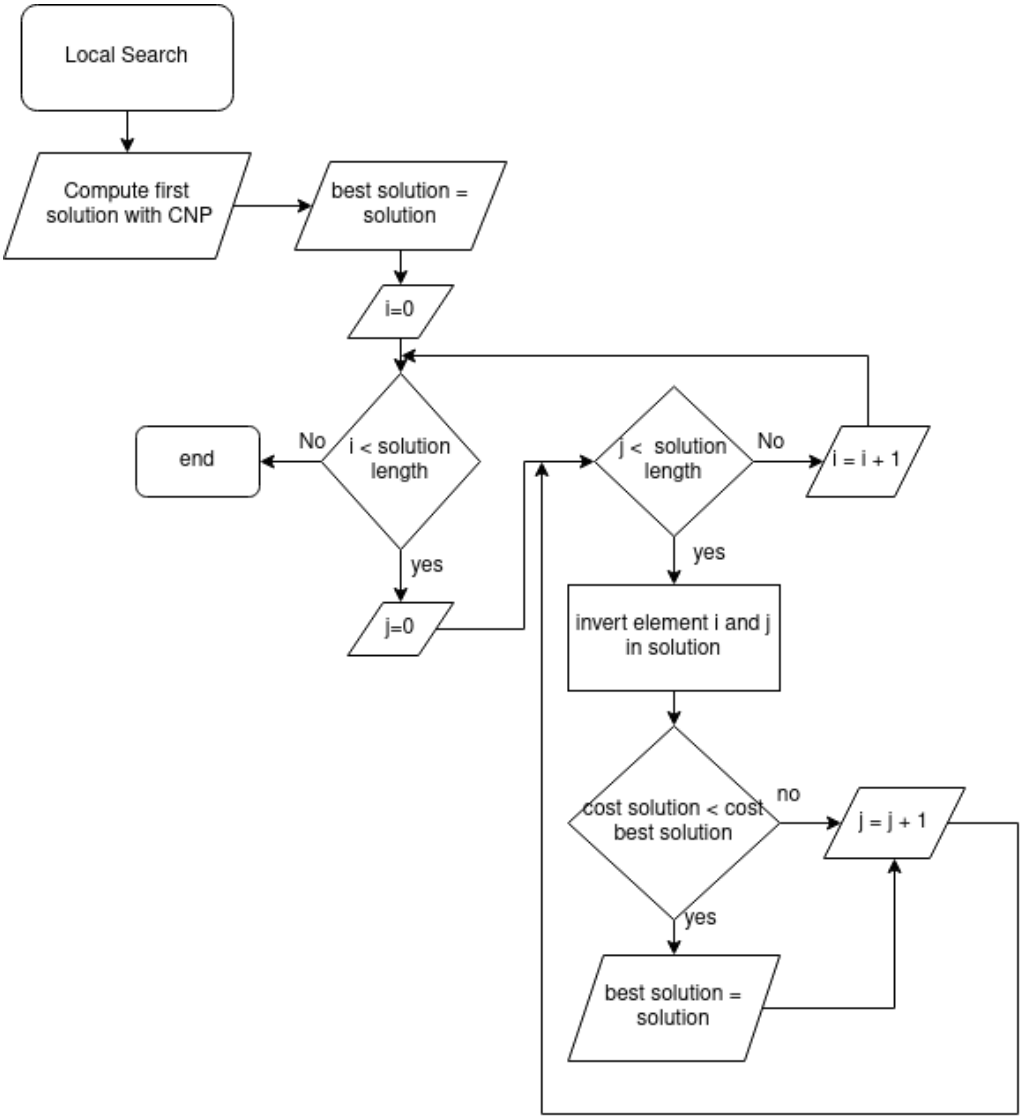


Figure 3.4 – Simplified Tabu Search flowchart.

3.4 Experimental results

In this section, we present a series of multi-robot task allocation results using the Tabu Search and Contract Net Protocol approach. The experiments that are done aim to find the best values for those criteria and obtain a function giving these values depending on some parameters such as the number of robots and the number of tasks. These experiments also compare the use of the TS algorithm with the CNP. With these comparisons, we can introduce a dynamic task allocator, which is able to switch between using CNP

and TS depending on different criteria. The objective is to minimize the traveling time of all the robots while having the smallest computational time possible. We consider the following equations to compare the allocation method with respect to the CNP.

Equation (3.5) expresses the overall execution time, which is an addition of the computation or allocation time of the algorithm and the traveling time of the robots to complete all tasks. Generally, centralized search algorithms tend to a give better solution, but they also require more computational time. This is why we are going to talk about the gain which will be the time gained from one solution to another taking into account both the computational time and the traveling time. The comparisons are always going to be done with the results obtained using CNP as a reference, as shown in Equation (3.6) and (3.7). Then the overall gain can be determined using Equation (3.8).

$$time_{experiment} = time_{traveling} + time_{computation} \quad (3.5)$$

$$gain_{traveling} = time_{traveling_{method}} - time_{traveling_{CNP}} \quad (3.6)$$

$$gain_{computation} = time_{computation_{method}} - time_{computation_{CNP}} \quad (3.7)$$

$$\begin{aligned} gain_{experiment} &= time_{experiment_{method}} - time_{experiment_{CNP}} \\ &= gain_{traveling} + gain_{computation} \end{aligned} \quad (3.8)$$

3.4.1 Experimental setups

Environment

The warehouse environment is simulated with a configuration as shown in Figure 3.5. Robots can only move in the white area, which represents free space. They can be at and start from any point of the map that is in the white area. The grey part represents walls, obstacles, or picking racks. Using this environment, the comparison of allocation algorithms includes a range between 2 and 10 robots, and up to 40 tasks.

Simulated robots

Good communication between vehicles, the ability to pick up and drop off objects, and good precision of the localization system are all assumed. The capacity to avoid obstacles or other robots is not taken into account in the simulation. Thus, no collision between vehicles is taken into account in the simulation. This ability or the notion of

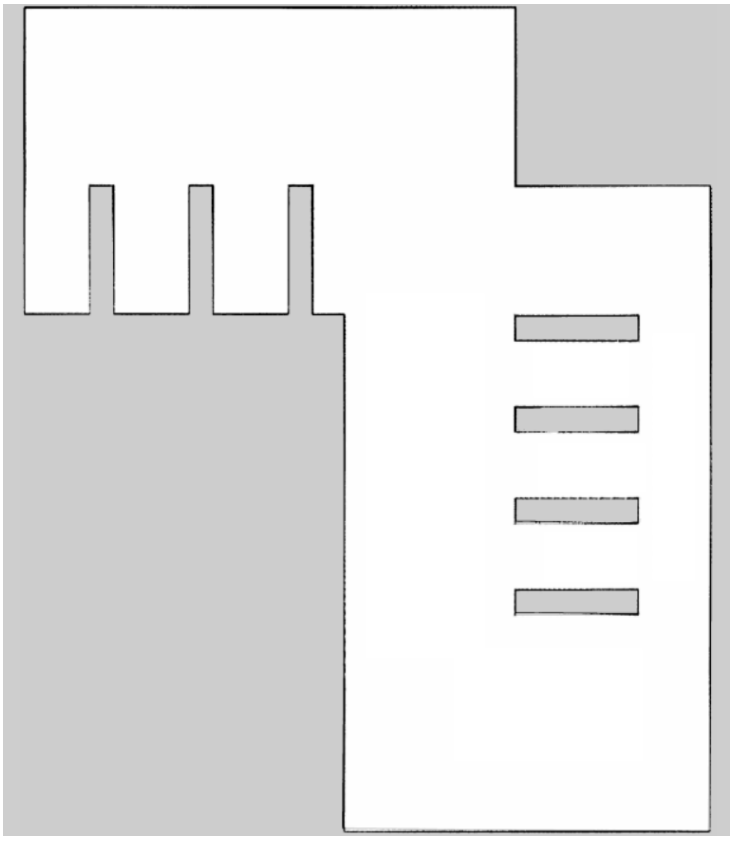


Figure 3.5 – 2D map of the simulated environment.

safety distances between the robots can be added. The robots can modify their velocity for this matter if needed. They are identical and take the same time to complete the tasks once they have reached the starting position of the task. The actual picking up and dropping off actions are not studied; it is assumed that they can perform them correctly. The robots that are not required for the tasks are idle. A list of assumptions is as follows:

1. Uniform acceleration,
2. Identical constant velocity v_{max},
3. Same skills,
4. Perfect control ,
5. Perfect localization,
6. Perfect Communication,
7. Infinite battery level.

Pre-estimated cost matrix

The values of estimated time to go from an ending point of a task to the starting point of another task are stored in a matrix to avoid recomputing them several times. The time estimation is based on a path that is generated by the A* path planning algorithm assuming there is no obstacle in the map. This path is then converted into time through an estimation of travel velocity. For algorithms such as TS, it appears that this estimated time is called many times in order to compute the traveling time of all neighbors of one solution. Therefore, the values of travel time from one task to another are the same as tasks to allocate do not change. For this reason, it is beneficial to pre-compute and stores these values in the memory. The time matrix is defined as follows:

$$\mathbf{M} = \begin{pmatrix} t_{1,1} & \cdots & t_{1,n} \\ \vdots & & \vdots \\ t_{n,1} & \cdots & t_{n,n} \end{pmatrix} \tag{3.9}$$

where $t_{i,i}$ is the time to travel task i from start to end and $t_{i,j}$ is the travel time from the endpoint of task i to the start point of task j.

In practice, the pre-computation can be done automatically by storing the estimated time to go from point A to B. If the same computation is required later on, then the cached value is recalled without any computation.

3.4.2 Preliminary experiments

Advantage of pre-computed cost values

To verify the advantage of using such matrix, an experiment is done comparing the computational time required for the allocation of 10 tasks to 5 robots through a TS algorithm with one iteration without storing the results of the estimate function in a cost matrix presented in Figure 3.6 and with storing the results of the estimate function in a cost matrix presented in Figure 3.7. Comparing those two figures it appears that thirty seconds per iteration of the TS algorithm can be gained from storing the estimated cost in the matrix. Indeed, each iteration of TS algorithm will call one time each value of the cost matrix. Therefore, computing the same cost each time is an imprudent use of the resource. A pre-computed cost matrix for tasks is used in the following experiments.

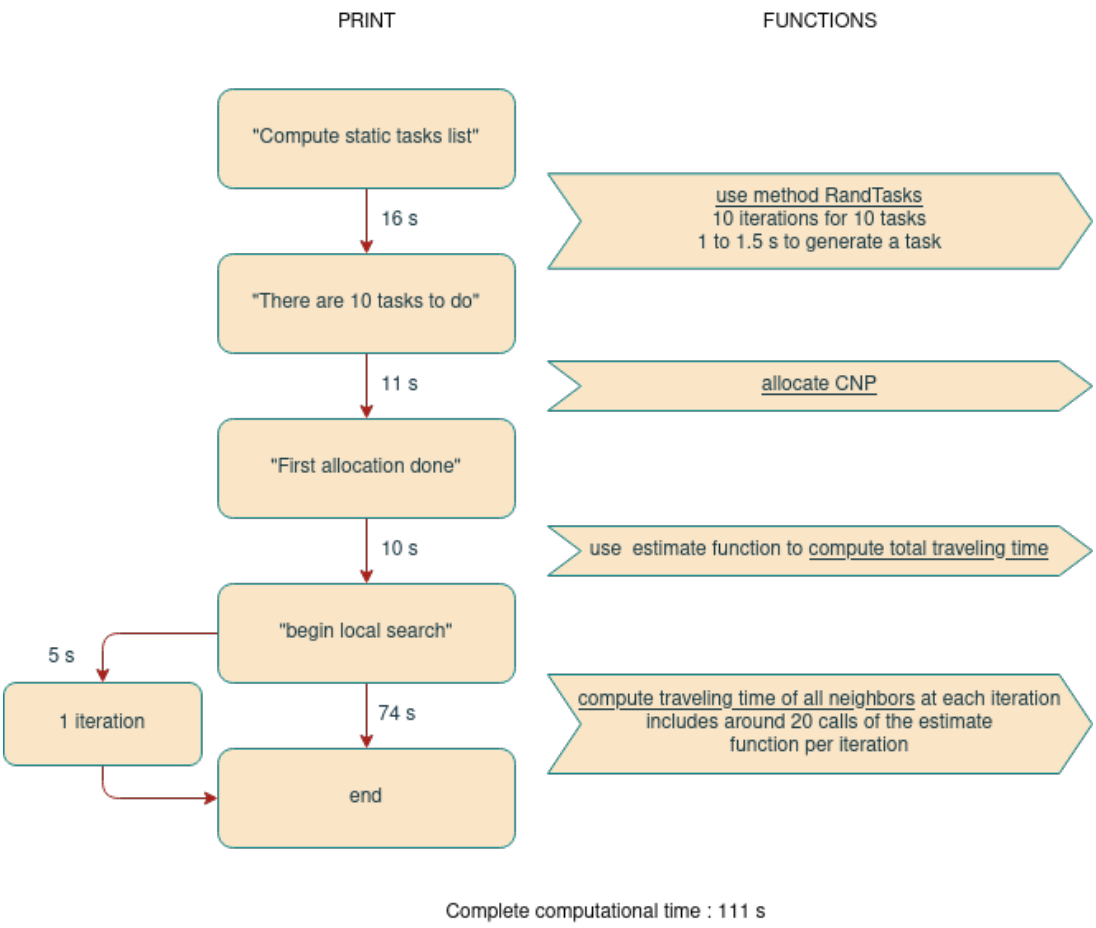


Figure 3.6 – Computational time required for the TS algorithm with one iteration without storing the results of the estimate function in a cost matrix

Comparison between Dijkstra and A* for path planning

It is possible to use several path-finding algorithms to select the best path taken by the robots and minimize the computational cost of the path-finding algorithm and the traveling distance of the robots. A comparison is made using the Dijkstra algorithm or A* algorithm in the allocation of 10 tasks to 5 robots.

- Results for 10 tasks, 5 robots using the CNP and Dijkstra algorithm:
  - Computation Time: 23s
  - Traveling Time: 1260s
- Results for 10 tasks, 5 robots using the CNP and A* algorithm:
  - Computation Time: 56s
  - Traveling Time: 1737s

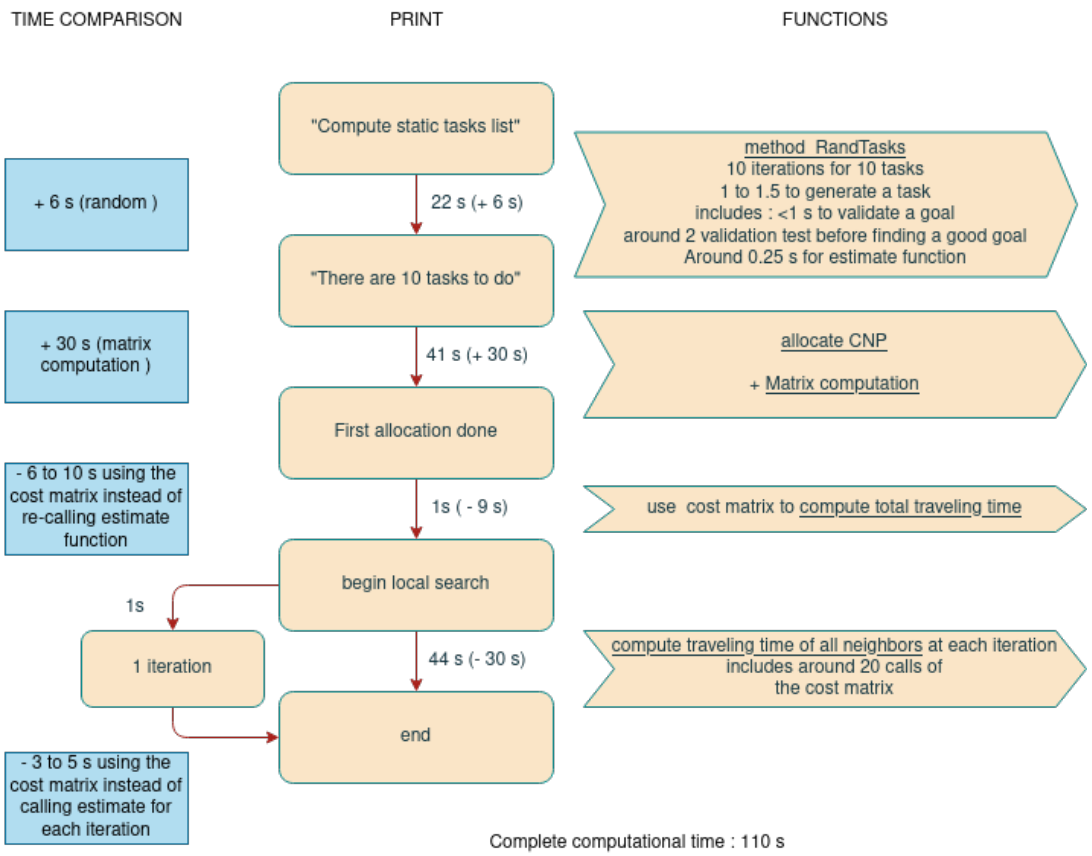


Figure 3.7 – Computational time required for the TS algorithm with one iteration with storing the results of the estimate function in a cost matrix

The impact of the choice of the path-finding algorithm is on the computational time of the estimate function and on the traveling distance of the robots. It appears that both the computational time and the traveling time increase using the A* algorithm for our simulation. Therefore, the Dijkstra algorithm is used in the following simulations.

Impact of the number of tasks on the computation and traveling time

The impact of the number of tasks on the computational time and on the traveling is studied for the case where 5 robots are available using either the CNP or the TS algorithm with 1, 2, 4, or 10 iterations. The impact of the number of tasks on the computational time is presented in Figure 3.8. The impact of the number of tasks on the traveling time is presented in Figure 3.9.

It can be observed in Figure 3.8 that for all allocation methods, the computational time increases with the number of tasks that have to be allocated. This increase in

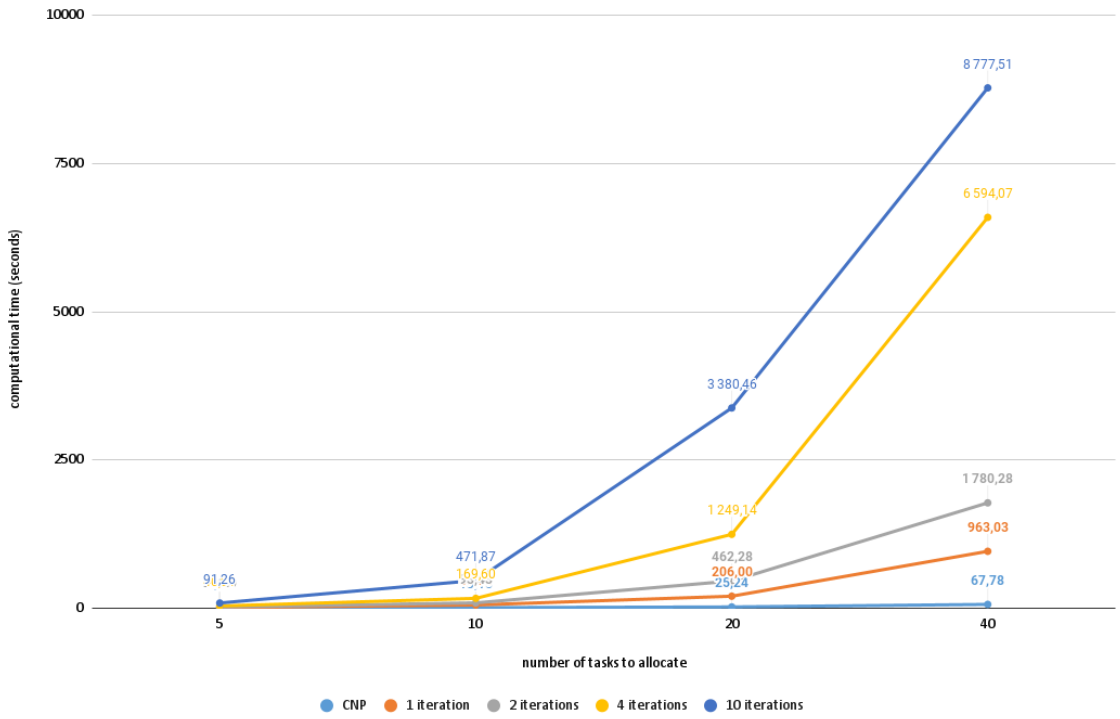


Figure 3.8 – Impact of the number of tasks to allocate to 5 robots on the computational time of the studied algorithms

computational time is faster for the TS algorithm than for the CNP, and it is faster when many iterations of the TS are done. Indeed, it can be observed that 91.26s are required to allocate 5 tasks and for 10 iterations of the TS algorithm, 8777.51s of computation are required to allocate 40 tasks with the same algorithm. This is an increase of 9519% of the time required for 5 tasks. When using the CNP 5.13s are required to allocate 5 tasks and 67.78s are required to allocate 40 tasks. This is an increase of 1220%. The explanation of the impact of the number of tasks on the allocation algorithms is the increasing number of operations that comes with the increasing number of tasks. This is greater for the TS algorithm as this algorithm calls a loop on the number of tasks inside a loop on the number of tasks which means it has approximately m^2 operations to do, where m is the number of tasks.

It can be observed in Figure 3.9 that for all allocation methods, the total traveling time increases with the number of tasks that are allocated. As an example, the total traveling time for the 5 robots is 2980.62s when they are doing 40 tasks that were allocated through

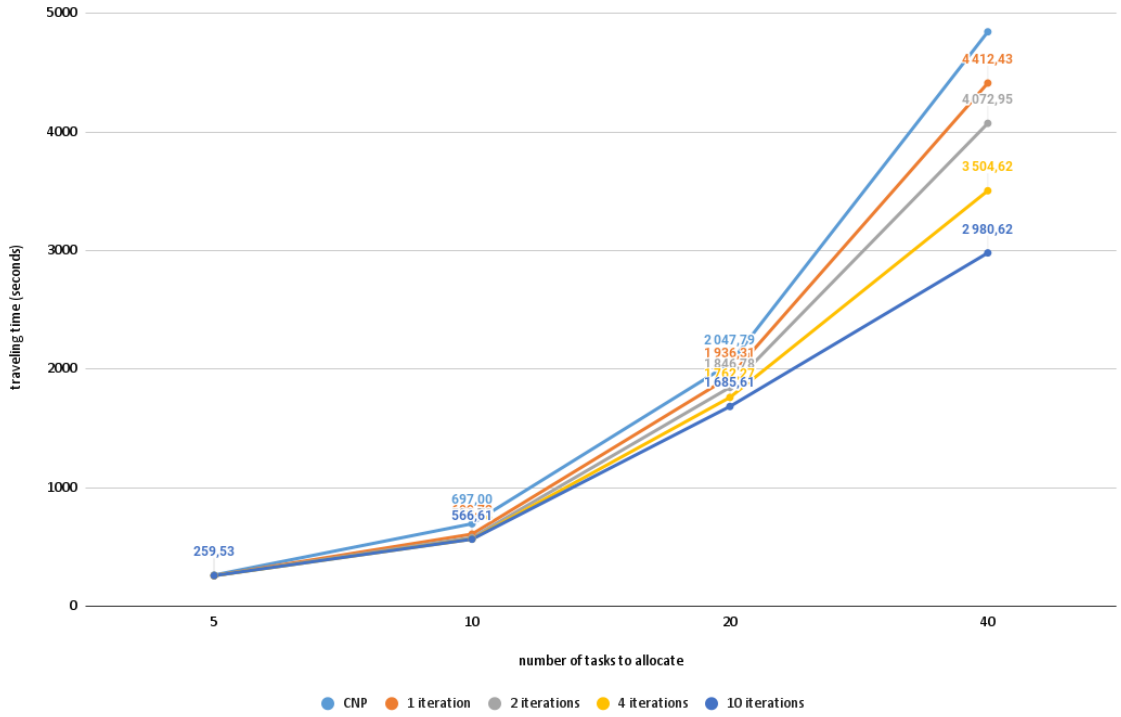


Figure 3.9 – Impact of the number of tasks allocated to 5 robots on the traveling time of the robots depending on the studied algorithms

a TS algorithm with 10 iterations and 259.53s when they are doing 5 tasks allocated through the same algorithm. It is 4844.45s for 40 tasks allocated through the CNP and 264.64s for 5 tasks allocated through the CNP. It can also be seen that the total traveling time is lower with the TS algorithm than with the CNP and when many iterations of the TS algorithm are used. Indeed, the solution that is found reduces the cost function value, although it requires more computational time as it was previously highlighted. This improvement of the cost function value is greater when a high number of tasks is allocated. As an example the improvement for the allocation of 5 tasks using the TS with 10 iterations compared to when using the CNP is 1.9%, it is 38% for the allocation of 40 tasks.

Impact of the number of robots on the computation and traveling time

The impact of the number of robots on the computational time and on the traveling time is studied for the case where 10 tasks are allocated to the robots using either the

CNP or the TS algorithm with 1, 2, 4, or 10 iterations. The impact of the number of robots on the computational time is presented in Figure 3.10. The impact of the number of robots on the traveling time is presented in Figure 3.11.

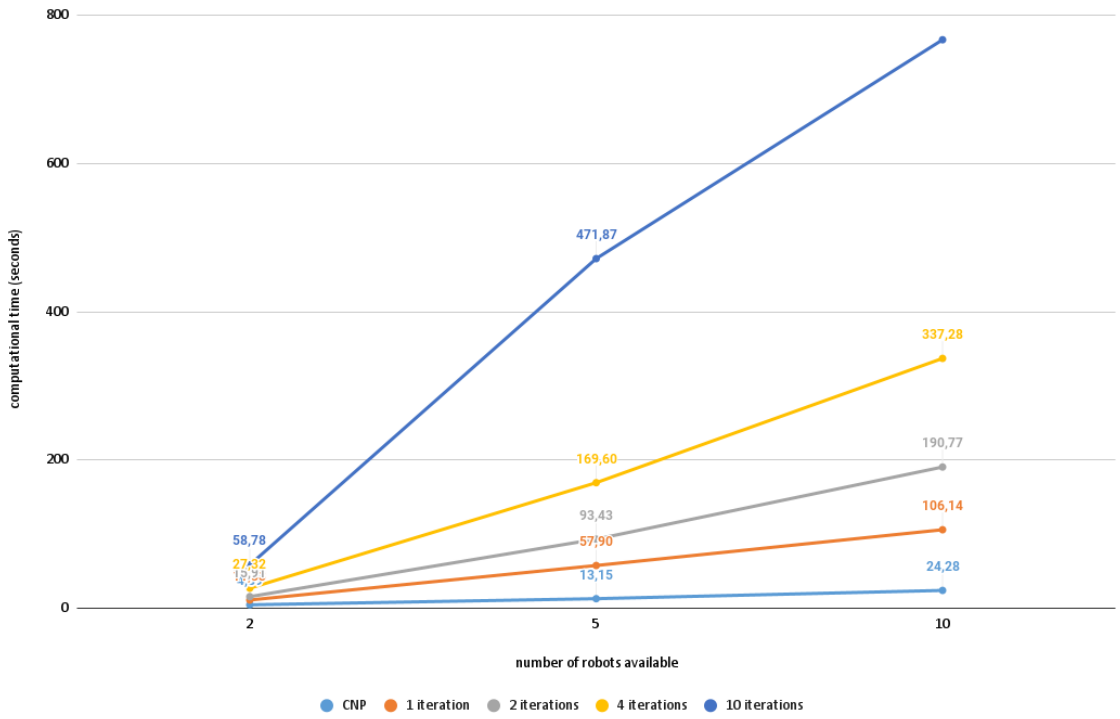


Figure 3.10 – Impact of the number of robots on the computational time of the studied algorithms for the allocation of 10 tasks

It can be observed in Figure 3.10 that for all allocation methods, the computational time increases with the number of robots that are used in the multi-robot system (MRS). This increase in computational time is faster for the TS algorithm than for the CNP, and it is faster when many iterations of the TS are done. Indeed, it can be observed that using the CNP, the computational time required for the allocation of 10 tasks to robots is 4.89s for 2 robots and 24.28s for 10 robots. This is an increase of 396%. Using 4 iterations of the TS algorithm this time is 27.32s for 2 robots and 337.28s for 10 robots. This is an increase of 1134%. Using 10 iterations of the TS algorithm this time is of 58.78s for 2 robots and of 767.12s for 10 robots. This is an increase of 1205%.

It can be observed in Figure 3.11 that for all allocation methods, the total traveling time decreases with the number of robots that are used in the multi-robot system (MRS).

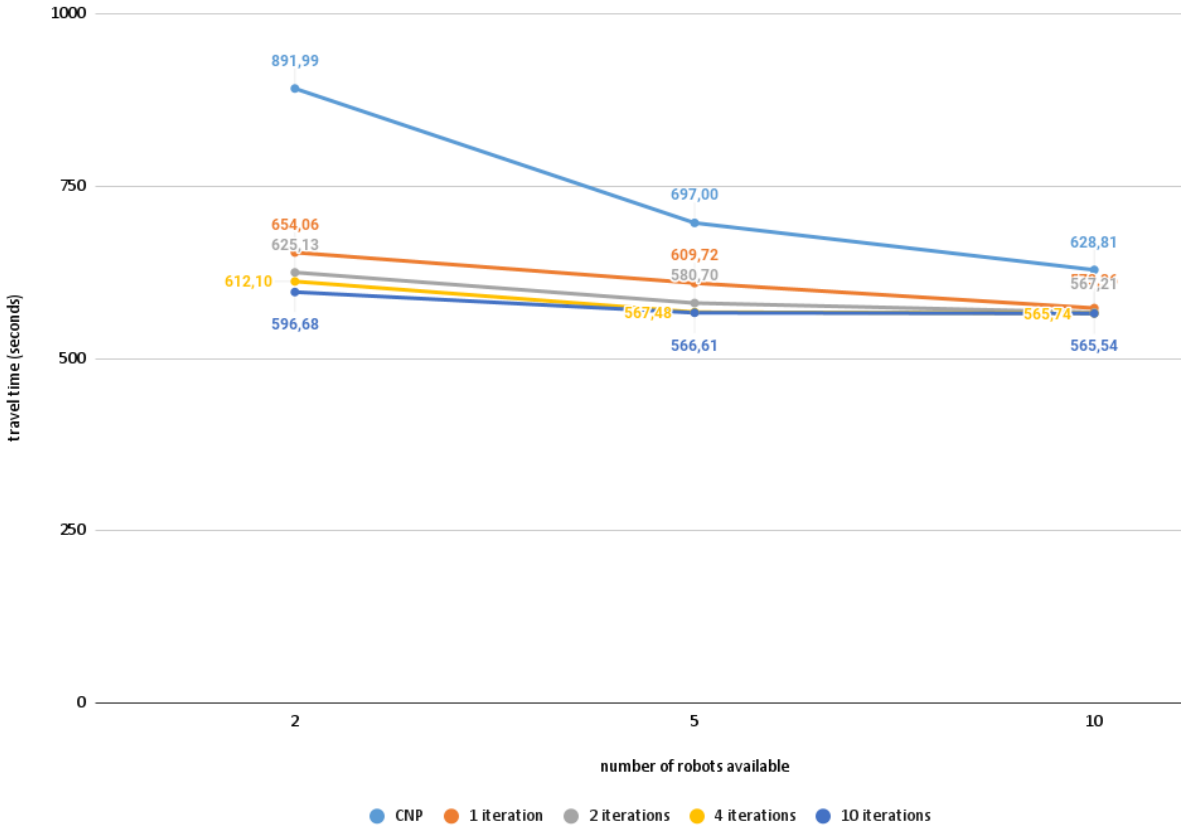


Figure 3.11 – Impact of the number of robots on their total traveling time using the studied algorithms for the allocation of 10 tasks

This decrease in total traveling time is faster for the CNP than for the TS algorithm, and it is slower when many iterations of the TS are done. Indeed, it can be observed that using the CNP, the total traveling time required to do the 10 tasks is 891.99s for 2 robots and 628.81s for 10 robots. This is a decrease of 29.5%. Using four iterations of the TS algorithm this time is 612.10s for 2 robots and 565.74s for 10 robots. This is a decrease of 8%. Using 10 iterations of the TS algorithm this time is 596.68s for 2 robots and 565.54s for 10 robots. This is a decrease of 5%. It is also observed that, although the decrease is slower for the TS algorithm than for the CNP, the total traveling time is always better when a high number of iterations is done and is better for the TS algorithm than for the CNP. But the more robots are in the MRS, the smallest this improvement is. There is also less improvement from doing one extra iteration of the TS algorithm than from switching from the CNP to the simplified Local Search method, or the TS algorithm with one iteration.

All in all, the more robots there are in the MRS, the less interesting it is to use the TS algorithm in terms of traveling time and computational time. However, the improvement of traveling time using the TS algorithm with one iteration compared to using the CNP is high, about 27% with a 2-robot-MRS and 10% with a 10-robot-MRS. In addition, the increase of the computational time is slow with the increase in the number of robots for one iteration of the TS. Therefore, only the Tabu Search algorithm with one iteration will be compared with the Contract Net Protocol in detail, which is presented in the next section.

3.4.3 Comparison experiments

5 tasks using CNP and Tabu Search algorithm

The first case studied is the allocation of 5 tasks. The results are presented in Figure 3.12. It can be observed that, in the case of the allocation of 5 tasks, the best method of allocation with MRS of 5 and 10 robots is the CNP.

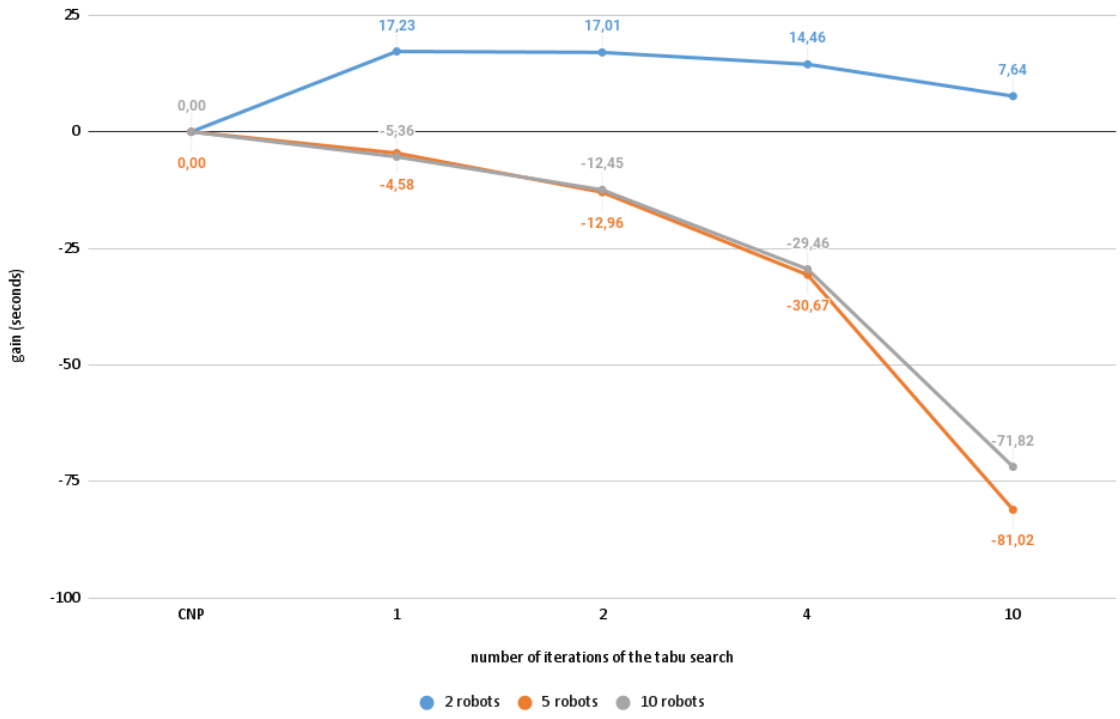


Figure 3.12 – Time gain evolution depending on the allocation mode and the number of robots for 5 tasks allocated.

Indeed, the gain is immediately negative with the use of the TS algorithm. It decreases even more with the increase in the number of iterations done. This is consistent with the results observed in Section 3.4.2 as the increase in the number of tasks implies an increase in the computational time that is faster for the TS algorithm than for the CNP, while the improvement seen in the value of the cost becomes the smallest when many iterations of the TS algorithm are done when the number of robots used in the MRS is high. Therefore, a high increase of the computation time for a low decrease of the traveling time when using the TS algorithm compared to the CNP for an increasing number of robots. This explains the decrease of the gain that comes with the increase of the number of robots. Moreover, it is seen in Section 3.4.2 that for a low number of tasks the amount of improvement in the traveling time gained from using the TS algorithm is even lower. This explains that in the case of the allocation of 5 tasks the best method is the CNP even when 5 robots are used in the MRS. However, for a MRS with 2 robots, the best method is the Tabu Search algorithm with one iteration. The gain observed for the allocation of 5 tasks to 2 robots using the TS algorithm with one iteration is of 17.2s compared to when using the CNP. This is due to the fact that for a low number of robots, the increase of the computation time using the TS is low and the benefit in the traveling time is high as seen in Section 3.4.2.

10 tasks using CNP and Tabu Search algorithm

With the number of tasks increased to 10, the allocation results are presented in Figure 3.13. It can be observed that for the use of 2 robots when 10 tasks are allocated, the gain is best for 4 iterations of the Tabu Search algorithm. It is 257.45s for 4 iterations so it corresponds to an improvement of 28.7% of the complete experiment time obtained through the CNP algorithm which is 896.9s. For the use of 5 robots when 10 tasks are allocated, the gain is best for one iteration of the Tabu Search algorithm, corresponding to the simplified Tabu Search algorithm. It is 42.54s for one iteration of the TS algorithm so it corresponds to an improvement of 6% of the complete experiment time obtained through the CNP algorithm, which is 710s. For the use of 10 robots when 10 tasks are allocated, there is no gain using the Tabu Search algorithm. It is best to only use the CNP. Indeed, as seen in Section 3.4.2 the more robots are in the MRS, the less improvement it is in terms of traveling time, and the more computational time is required.

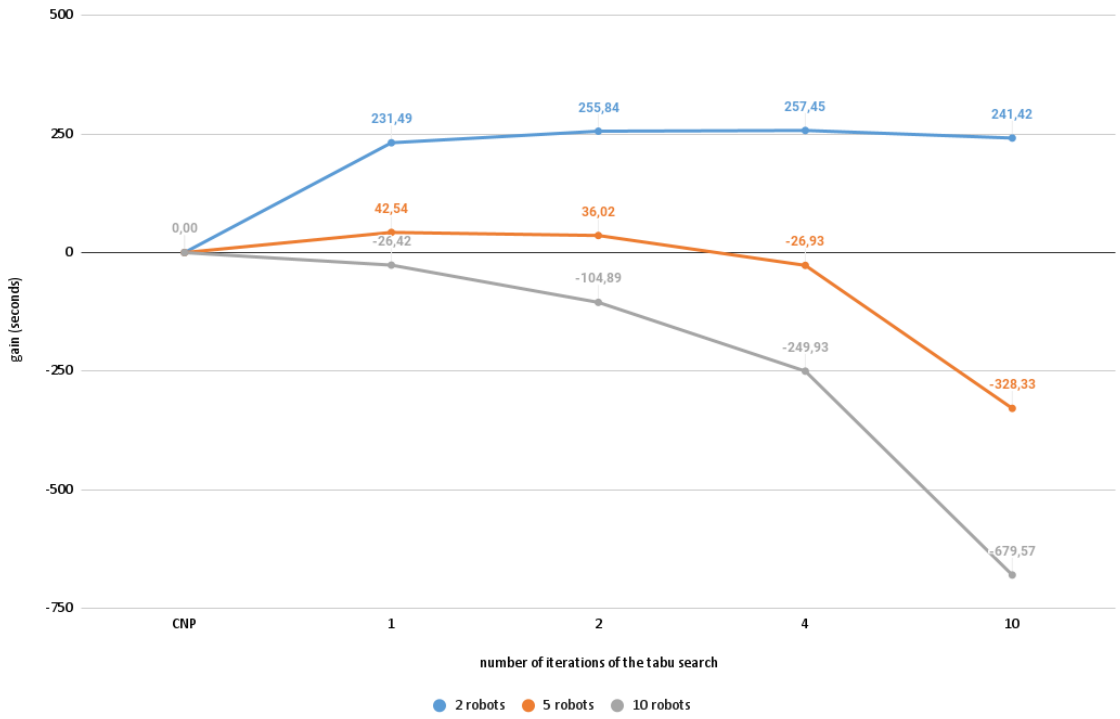


Figure 3.13 – Time gain evolution depending on the allocation mode and the number of robots for 10 tasks allocated.

20 Tasks using CNP and Tabu Search algorithm

The third case of this experiment is for the allocation of 20 tasks. The results are presented in Figure 3.14. As seen in the two previous cases the gain is decreasing with the number of robots in the MRS and with the number of iterations of the TS algorithm. Oppositely to before the gain is very low even for the use of only 2 robots in the MRS. This is due to the impact of the number of tasks to allocate seen in Section 3.4.2. Indeed, the more tasks are added, the more quickly the computational time increases with the number of iterations. In the case of the allocation of 20 tasks, the gain is slightly positive when using one iteration of the TS algorithm for a 2-robot-MRS. It is 56.50s, which corresponds to an improvement of 3% of the total execution time when the CNP is used which is 1760s. For MRS of more robots, the CNP is the best method to use.

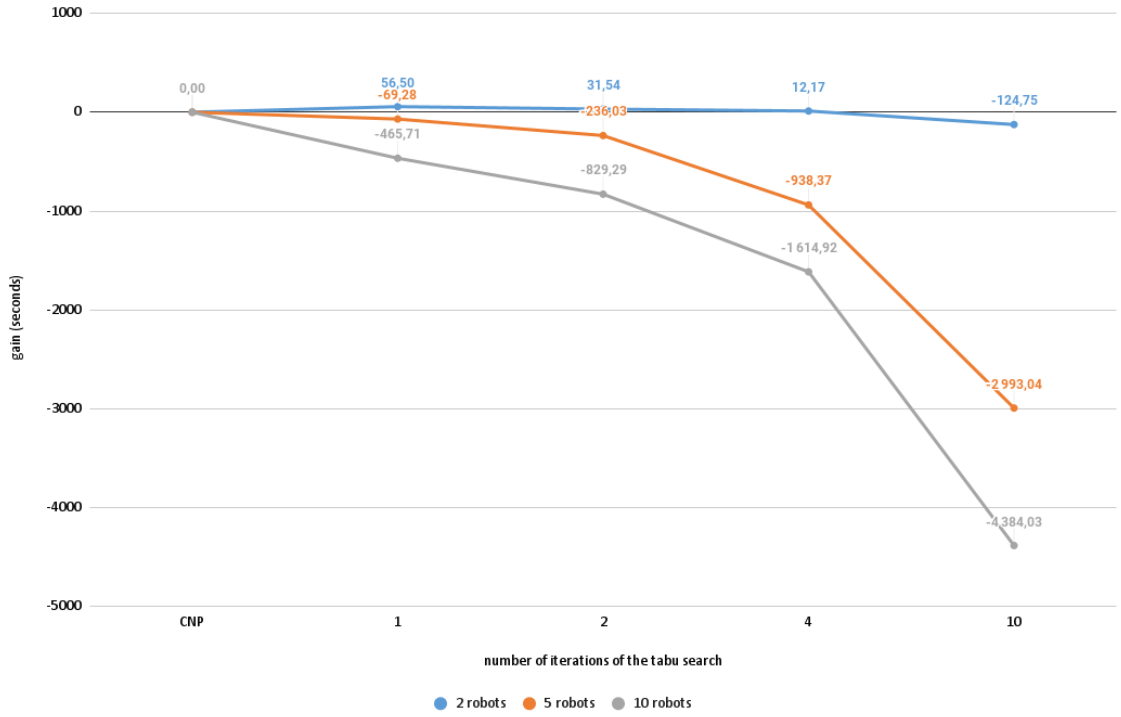


Figure 3.14 – Time gain evolution depending on the allocation mode and the number of robots for 20 tasks allocated.

40 Tasks using CNP and Tabu Search algorithm

Last but not least, an allocation of 40 tasks is simulated, whose results are presented in Figure 3.15. The results are similar to the one presented in Figure 3.14 for 20 tasks allocated. The gain is again decreasing with the number of robots and with the number of iterations. The gain is also low for the 2-robot-MRS. As previously explained, this is due to the fact that the more tasks are added the more quickly the computational time increases with the number of iterations as seen in Section 3.4.2. In the case of the allocation of 40 tasks, the gain is slightly positive when using the TS algorithm for a 2-robot-MRS. It is best with 4 iterations of this algorithm. It is 724.10s in this case, which corresponds to an improvement of 14.5% of the total execution time when the CNP is used which is 4987s. For a MRS of more robots, CNP is still the best method to use.

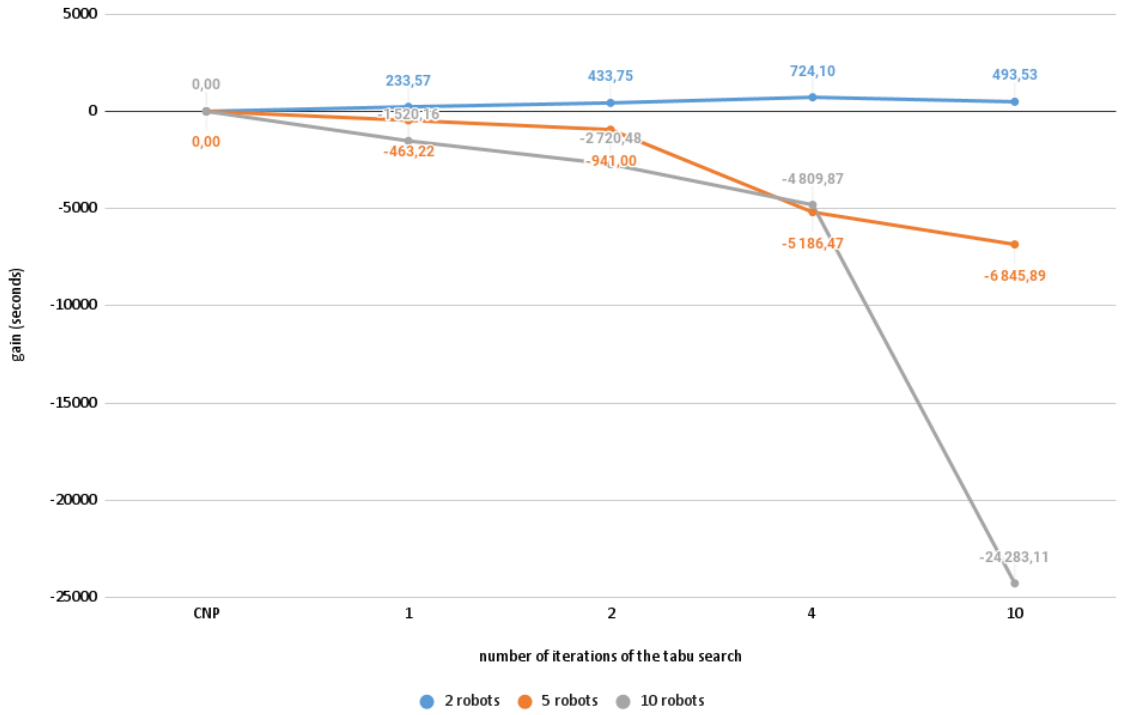


Figure 3.15 – Time gain evolution depending on the allocation mode and the number of robots for 40 tasks allocated.

		Number of robots		
		2	5	10
Number of Tasks	5	TS-1 (6% gain)	CNP	CNP
	10	TS-4 (29% gain)	TS-1 (6% gain)	CNP
	20	TS-1 (3% gain)	CNP	CNP
	40	TS-4 (15% gain)	CNP	CNP

Figure 3.16 – Best results of different cases of robots and tasks.

The results of which algorithm is best for which case for our MRTA problem are presented in the table in Figure 3.16. It summarizes all the results obtained in Section 3.4.3. For MRS with 5 or more robots, the CNP is the best allocation method. The gain ob-

tained with the use of the TS algorithm is either negative or too low to be generalized as being indeed positive with another computing system. However, for MRS with only two robots, the use of the TS algorithm is interesting, although one iteration of the TS algorithm is often enough to benefit from a better allocation compared to CNP. Since it is rarely the case for a warehouse to have less than 5 robots running, the search algorithm such as TS is not suitable for our application.

3.5 Conclusion on task allocation

A fundamental principle of task allocation, formulation of the objective function, is discussed in this chapter. We make a performance comparison between two task allocation approaches, namely Contract Net Protocol and Tabu Search. The comparison focuses on transportation tasks, which respond to the actual needs of a warehouse. The comparison criteria chosen is based on the overall total time, which includes the time needed to allocate and to complete a task. In addition, we show that having a pre-estimated cost matrix on travel time can be a benefit in terms of time-saving measure during the allocation process. Tasks are in form of a static list given at the start of allocation and dynamic, which are not pre-defined; they are added online during the runtime. As the number of robots in the environment increases, CNP takes significantly less time to perform the allocation compared to the TS. Therefore, CNP is more suitable for our application.

Conclusion

Despite the latest advancements and research on formation techniques and the fact that the multi-robot system is more flexible and efficient in terms of huge load transportation tasks, we are yet to see an actual implementation of such systems in real-world environments. This thesis's objective is to bridge this gap between laboratory research and the industry. Three research axes have been conducted:

— Consensus-based formation control

— Optimization-based formation control

— Determination of task allocation approach for warehouse application

Regarding the first pole of research, it is shown that a fleet of nonholonomic robots is able to safely navigate the environment without breaking the formation. Consensus-based formation control is a distributed controller, which opens up the possibility of having a huge number of robots in a fleet. On the other hand, due to the nature of consensus, this approach can not ensure precisely the desired formation as the controller itself is made up of several weighted components (formation, navigation, obstacle avoidance, etc.), which may dominate each other in various scenarios resulting in undesired behaviors.

The optimization-based controller is proposed to solve the mentioned issue. Each control component is defined as a task with a different priority to be respected by the solver. With task priority introduced, a two-level optimization problem is formed and can be solved using hierarchical quadratic programming. This approach shows better performance in terms of being able to keep the formation. In addition to being able to achieve precisely the formation shape and desired inter-distance between robots, we can also control the heading angle of the whole fleet, which improves the navigation behavior.

Simulations and experiments on Huskys provided by E-COBOT are conducted for the two proposed formation algorithms. In the case of the optimization-based controller, we have shown that a sophisticated connection mechanism between the load and the robots is not required. It is because of the desired inter-distances that are well-kept throughout the navigation.

The third pole of this work shows the interest in using the Contract Net Protocol over the Tabu Search for task allocation problems in a simulated warehouse environment with

a different number of tasks and robots. Due to the distributed nature of the CNP, it can allocate tasks quickly but is not necessarily optimal. By contrast, even though the objective cost of TS is always equal to or lower than that of CNP, its computational time required is too high compared to that of CNP.

Despite having favorable positive results in this thesis, several potential perspectives for future work can be considered. First of all, the current implementation of the proposed consensus formation control does not include the orientation control of the fleet's heading; future studies could extend the approach to include this part, which could improve navigation and obstacle avoidance behavior. For the optimization-based controller, it is interesting to extend to work to a distributed manner similar to what has been studied for the null-space approach in [Tru+18]. The authors propose a controller that consists of a local task for obstacle avoidance and a global task for the formation. The individual avoidance task has more priority than the consensus formation control.

Another interesting extension to the formation control as a whole is to include trajectory planning in the multi-robot system. Since the current controllers are the reactive ones, a risk of getting stuck in a local minimum due to the obstacle's shape is prominent. A study of a trajectory planner that considers the geometric shape and constraint of the fleet can solve this issue. Thus, the navigation task of the controllers will just need to track waypoints generated by such a planner.

With recent parallel computing technology, it is compelling to study the improvement parallel computing makes to the Tabu Search algorithm in terms of computational time. In addition, the accuracy of the robot's traveling time estimation can be improved by having feedback on the actual travel time to adapt the estimation process. Last but not least, adding collision avoidance and multi-robot path planning could extend the accuracy of time estimation and increase the overall system's efficiency.

BIBLIOGRAPHY

[ABR17] Javier Alonso-Mora, Stuart Baker, and Daniela Rus, « Multi-robot formation control and object transport in dynamic environments via constrained optimization », *in*: *The International Journal of Robotics Research* 36.*9* (Aug. 2017), pp. 1000–1021 (cit. on pp. 23, 24).

[AFH14] Shakeel Ahmad, Zhi Feng, and Guoqiang Hu, « Multi-robot formation control using distributed null space behavioral approach », *in*: *Proceedings - IEEE International Conference on Robotics and Automation* (2014), pp. 3607–3612 (cit. on p. 21).

[AMI89] H. Asama, A. Matsumoto, and Y. Ishida, « Design Of An Autonomous And Distributed Robot System: Actress », *in*: *Proceedings. IEEE/RSJ International Workshop on Intelligent Robots and Systems '. (IROS '89) 'The Autonomous Mobile Robots and Its Applications*, Tsukuba: IEEE, 1989, pp. 283–290 (cit. on p. 16).

[ANT17] Muhanad H.Mohammed Alkilabi, Aparajit Narayan, and Elio Tuci, « Cooperative object transport with a swarm of e-puck robots: robustness and scalability of evolved collective strategies », *in*: *Swarm Intelligence* 11 (2017), pp. 185–209 (cit. on p. 24).

[Bar+20] Uthman Baroudi et al., « Dynamic Multi-Objective Auction-Based (DYMO-Auction) Task Allocation », *in*: *Applied Sciences* 10.*9* (May 2020), p. 3264 (cit. on p. 26).

[BCD19] F. Basile, P. Chiacchio, and E. Di Marino, « An auction-based approach to control automated warehouses using smart vehicles », *in*: *Control Engineering Practice* 90.*June* (Sept. 2019), pp. 285–300 (cit. on pp. 26, 76).

[Bro86] R. Brooks, « A robust layered control system for a mobile robot », *in*: *IEEE Journal on Robotics and Automation* 2.*1* (1986), pp. 14–23 (cit. on p. 21).

[BSM20] Ali B. Bahgat, Omar M. Shehata, and El Sayed I. Morgan, « A Multi-Level Architecture for Solving the Multi-Robot Task Allocation Problem Using a Market-Based Approach », *in*: *International Journal of Mechanical Engineering and Robotics Research* 9.*2* (2020), pp. 293–298 (cit. on p. 26).

[BWN09] Thijs H.A. van den Broek, Nathan van de Wouw, and Henk Nijmeijer, « Formation control of unicycle mobile robots: a virtual structure approach », *in*: *Proceedings of the 48h IEEE Conference on Decision and Control (CDC) held jointly with 2009 28th Chinese Control Conference*, Shanghai, China, Dec. 2009, pp. 8328–8333 (cit. on p. 20).

[Cal+90] P. Caloud et al., « Indoor automation with many mobile robots », *in*: *IEEE International Workshop on Intelligent Robots and Systems, Towards a New Frontier of Applications*, Ibaraki: IEEE, July 1990, pp. 67–72 (cit. on p. 16).

[CB15] Lei Chen and Ma Baoli, « A nonlinear formation control of wheeled mobile robots with virtual structure approach », *in*: *2015 34th Chinese Control Conference*, vol. 2015-Septe, Hangzhou, China, July 2015, pp. 1080–1085 (cit. on p. 20).

[Che+15] Jianing Chen et al., « Occlusion-Based Cooperative Transport with a Swarm of Miniature Mobile Robots », *in*: *IEEE Trans.on Robotics* 31 (Apr. 2015), pp. 307–321 (cit. on p. 24).

[Con+06] Luca Consolini et al., « On the Control of a Leader-Follower Formation of Nonholonomic Mobile Robots », *in*: *Proceedings of the 45th IEEE Conference on Decision and Control*, San Diego, CA, USA: IEEE, Dec. 2006, pp. 5992–5997 (cit. on p. 19).

[Dai+16] Yanyan Dai et al., « Symmetric caging formation for convex polygonal object transportation by multiple mobile robots based on fuzzy sliding mode control », *in*: *ISA Transactions* 60 (2016), pp. 321–332 (cit. on p. 24).

[Die05] Reinhard Diestel, *Graph Theory*, 2005 (cit. on p. 35).

[Eoh+11] Gyuho Eoh et al., « Multi-robot cooperative formation for overweight object transportation », *in*: *2011 IEEE/SICE International Symposium on System Integration, SII 2011 c* (2011), pp. 726–731 (cit. on pp. 23, 24).

[Fal+11] Riccardo Falconi et al., *A graph-based collision-free distributed formation control strategy*, vol. 44, 1 PART 1, IFAC, 2011, pp. 6011–6016 (cit. on p. 32).

[FH08] Bans Fidan and Julien M. Hendrickx, « Rigid graph control architectures for autonomous formations », *in*: *IEEE Control Systems* 28.*6* (Dec. 2008), pp. 48–63 (cit. on p. 37).

[Fu+19] Junjie Fu et al., « Consensus of second-order multiagent systems with both velocity and input constraints », *in*: *IEEE Transactions on Industrial Electronics* 66.*10* (2019), pp. 7946–7955 (cit. on p. 22).

[G L+05] Michail G. Lagoudakis et al., « Auction-Based Multi-Robot Routing », *in*: *Robotics: Science and Systems I*, vol. 1, Robotics: Science and Systems Foundation, June 2005, pp. 343–350 (cit. on p. 25).

[Gif+10] Christopher M. Gifford et al., « A novel low-cost, limited-resource approach to autonomous multi-robot exploration and mapping », *in*: *Robotics and Autonomous Systems* 58.*2* (Feb. 2010), pp. 186–202 (cit. on p. 11).

[GM04] Brian P. Gerkey and Maja J. Matarić, « A Formal Analysis and Taxonomy of Task Allocation in Multi-Robot Systems », *in*: *The International Journal of Robotics Research* 23.*9* (Sept. 2004), pp. 939–954 (cit. on p. 24).

[GM12] Avinash Gautam and Sudeept Mohan, « A review of research in multi-robot systems », *in*: *2012 IEEE 7th Int. Conf. on Industrial and Information Systems (ICIIS)*, Chennai, India, Aug. 2012 (cit. on p. 11).

[Hou+17] Zhicheng Hou et al., « A survey on the formation control of multiple quadrotors », *in*: *2017 14th International Conference on Ubiquitous Robots and Ambient Intelligence (URAI)*, Jeju, South Korea, June 2017, pp. 219–225 (cit. on pp. 18, 22, 32).

[Hus+18] Ahmed Hussein et al., « Hybrid Optimization-Based Approach for Multiple Intelligent Vehicles Requests Allocation », *in*: *Journal of Advanced Transportation* 2018 (2018), pp. 1–11 (cit. on p. 12).

[KC14] Olivier Kermorgant and François Chaumette, « Dealing with constraints in sensor-based robot control », *in*: *IEEE Trans. on Robotics* (Feb. 2014), pp. 244–257 (cit. on p. 50).

[KHE15] Alaa Khamis, Ahmed Hussein, and Ahmed Elmogy, « Multi-robot Task Allocation: A Review of the State-of-the-Art », *in*: *Cooperative Robots and Sensor Networks 2015*, vol. 604, 2015, pp. 31–51 (cit. on pp. 25, 26).

[KLW11] Oussama Kanoun, Florent Lamiraux, and Pierre Brice Wieber, « Kinematic control of redundant manipulators: Generalizing the task-priority framework to inequality task », *in*: *IEEE Transactions on Robotics* 27.*4* (2011), pp. 785–792 (cit. on p. 51).

[KSD13] G Ayorkor Korsah, Anthony Stentz, and M Bernardine Dias, « A comprehensive taxonomy for multi-robot task allocation », *in*: *The International Journal of Robotics Research* 32.*12* (Oct. 2013), pp. 1495–1512 (cit. on p. 24).

[LCS15] Lingzhi Luo, Nilanjan Chakraborty, and Katia Sycara, « Distributed Algorithms for Multirobot Task Assignment With Task Deadline Constraints », *in*: *IEEE Transactions on Automation Science and Engineering* 12.*3* (July 2015), pp. 876–888 (cit. on p. 26).

[LHW14] Jin Ling Lin, Kao Shing Hwang, and Ya Ling Wang, « A Simple Scheme for Formation Control Based on Weighted Behavior Learning », *in*: *IEEE Transactions on Neural Networks and Learning Systems* 25.*6* (June 2014), pp. 1033–1044 (cit. on p. 22).

[Liu+17] Andong Liu et al., « Nash-optimization distributed model predictive control for multi mobile robots formation », *in*: *Peer-to-Peer Networking and Applications* 10.*3* (May 2017), pp. 688–696 (cit. on p. 22).

[LLL15] Jing Luo, Cheng-Lin Liu, and Fei Liu, « A leader-following formation control of multiple mobile robots with obstacle », *in*: *2015 IEEE International Conference on Information and Automation*, 61473138, Lijiang, China, Aug. 2015, pp. 2153–2158 (cit. on p. 19).

[LN11] Chang Boon Low and Quee San Ng, « A flexible virtual structure formation keeping control for fixed-wing UAVs », *in*: *2011 9th IEEE International Conference on Control and Automation (ICCA)*, Santiago, Chile, Dec. 2011, pp. 621–626 (cit. on p. 20).

[LRV14] Gilbert Laporte, Stefan Ropke, and Thibaut Vidal, « Chapter 4: Heuristics for the Vehicle Routing Problem », *in*: *Vehicle Routing*, vol. 7, 4-5,

Philadelphia, PA: Society for Industrial and Applied Mathematics, Nov. 2014, pp. 87–116 (cit. on pp. 12, 77).

[Mar+13] Alessandro Marino et al., « A Decentralized Architecture for Multi-Robot Systems Based on the Null-Space-Behavioral Control with Application to Multi-Robot Border Patrolling », *in*: *Journal of Intelligent & Robotic Systems* 71 (Sept. 2013), pp. 423–444 (cit. on p. 21).

[Mia+18] Zhiqiang Miao et al., « Distributed Estimation and Control for Leader-Following Formations of Nonholonomic Mobile Robots », *in*: *IEEE Transactions on Automation Science and Engineering* 15.*4* (Oct. 2018), pp. 1946–1954 (cit. on pp. 11, 18).

[MM06] Alejandro R. Mosteo and Luis Montano, « Simulated annealing for multi-robot hierarchical task allocation with flexible constraints and objective functions », *in*: *Workshop on Network Robot Systems: Toward Intelligent Robotic Systems Integrated with Environments at IROS*, Jan. 2006 (cit. on p. 27).

[NMS15] K. Noack, L. Marsh, and S. Shekh, « Negotiation Protocol Comparison for task allocation in highly dynamic environments », *in*: *Weber, T., McPhee, M.J. and Anderssen, R.S. (eds) MODSIM2015, 21st International Congress on Modelling and Simulation*, Modelling, Simulation Society of Australia, and New Zealand, Nov. 2015, pp. 718–724 (cit. on p. 12).

[Olf06] Reza Olfati-Saber, « Flocking for Multi-Agent Dynamic Systems: Algorithms and Theory », *in*: *IEEE Transactions on Automatic Control* 51.*3* (Mar. 2006), pp. 401–420 (cit. on pp. 21, 32, 39–41, 44, 45).

[PAM20] Héctor M. Pérez-Villeda, Gustavo Arechavaleta, and América Morales-Díaz, « Multi-vehicle coordination based on hierarchical quadratic programming », *in*: *Control Engineering Practice* 94 (Jan. 2020) (cit. on pp. 22, 24, 55).

[Pop+17] Mihai-Ioan Popescu et al., « Multi-cycle Coverage for Multi-robot Patrolling - application to data collection in WSNs - », *in*: *Journées Francophones sur la Planification, la Décision et l'Apprentissage pour la conduite de systèmes (JFPDA)*, Caen, France, July 2017 (cit. on p. 11).

[RB08] Wei Ren and Randal W Beard, *Distributed Consensus in Multi-vehicle Cooperative Control*, London: Springer London, 2008, pp. 198–199 (cit. on p. 33).

[Ren+10] Rui Ren et al., « Automatic generation of optimally rigid formations using decentralized methods », *in*: *International Journal of Automation and Computing* 7.*4* (Nov. 2010), pp. 557–564 (cit. on p. 36).

[Rey87] Craig W Reynolds, « Flocks, herds and schools: A distributed behavioral model », *in*: *Computer Graphics* 21.*4* (July 1987), pp. 25–34 (cit. on p. 21).

[See+20] N. Seenu et al., « Review on state-of-the-art dynamic task allocation strategies for multiple-robot systems », *in*: *Industrial Robot* 47.*6* (2020), pp. 929–942 (cit. on pp. 25, 26).

[SFZ19] Osamah Saif, Isabelle Fantoni, and Arturo Zavala-Río, « Distributed integral control of multiple UAVs: precise flocking and navigation », *in*: *IET Control Theory & Applications* 13.*13* (Sept. 2019), pp. 2008–2017 (cit. on pp. 21, 32, 39, 41).

[SGC19] Nick Sullivan, Steven Grainger, and Ben Cazzolato, « Sequential single-item auction improvements for heterogeneous multi-robot routing », *in*: *Robotics and Autonomous Systems* 115 (May 2019), pp. 130–142 (cit. on p. 26).

[SPB18] David St-Onge, Carlo Pinciroli, and Giovanni Beltrame, « Circle Formation with Computation-Free Robots Shows Emergent Behavioural Structure », *in*: *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Madrid, Spain, Oct. 2018, pp. 5344–5349 (cit. on p. 21).

[SR15] Kaarthik Sundar and Sivakumar Rathinam, « An exact algorithm for a heterogeneous, multiple depot, multiple traveling salesman problem », *in*: *2015 International Conference on Unmanned Aircraft Systems, ICUAS 2015* (2015) (cit. on p. 27).

[TAA18] Elio Tuci, Muhanad H. M. Alkilabi, and Otar Akanyeti, « Cooperative Object Transport in Multi-Robot Systems: A Review of the State-of-the-Art », *in*: *Frontiers in Robotics and AI* 5 (May 2018) (cit. on p. 23).

[Tru+18] Miguel A. Trujillo et al., « Priority Task-Based Formation Control and Obstacle Avoidance of Holonomic Agents with Continuous Control Inputs », *in*: *IFAC-PapersOnLine* 51.*13* (2018), pp. 216–222 (cit. on p. 98).

[Tsa+18] Kam Fai Elvis Tsang et al., « A Novel Warehouse Multi-Robot Automation System with Semi-Complete and Computationally Efficient Path Planning and Adaptive Genetic Task Allocation Algorithms », *in*: *2018 15th International Conference on Control, Automation, Robotics and Vision (ICARCV)*, IEEE, Nov. 2018, pp. 1671–1676 (cit. on p. 27).

[VR21] Janardan Kumar Verma and Virender Ranga, « Multi-Robot Coordination Analysis, Taxonomy, Challenges and Future Scope », *in*: *Journal of Intelligent & Robotic Systems* 102.*1* (May 2021), p. 10 (cit. on p. 16).

[Wan+17a] Hesheng Wang et al., « Adaptive Vision-Based Leader–Follower Formation Control of Mobile Robots », *in*: *IEEE Transactions on Industrial Electronics* 64.*4* (Apr. 2017), pp. 2893–2902 (cit. on p. 19).

[Wan+17b] Zhuping Wang et al., « A graph based formation control of nonholonomic wheeled robots using a novel edge-weight function », *in*: *2017 IEEE International Conference on Systems, Man, and Cybernetics, SMC 2017*, vol. 2017-Janua, 91420103, 2017, pp. 1477–1481 (cit. on p. 22).

[Wan+18] Yuanzhe Wang et al., « A Practical Leader–Follower Tracking Control Scheme for Multiple Nonholonomic Mobile Robots in Unknown Obstacle Environments », *in*: *IEEE Transactions on Control Systems Technology* 27.*4* (July 2018), pp. 1685–1693 (cit. on p. 19).

[Wan+19] Jia Wang et al., « Pattern-RL: Multi-robot Cooperative Pattern Formation via Deep Reinforcement Learning », *in*: *2019 18th IEEE International Conference On Machine Learning And Applications (ICMLA)*, IEEE, Dec. 2019, pp. 210–215 (cit. on p. 22).

[Wan+20] Weiwei Wan et al., « Multirobot Object Transport via Robust Caging », *in*: *IEEE Transactions on Systems, Man, and Cybernetics: Systems* 50.*1* (Jan. 2020), pp. 270–280 (cit. on pp. 23, 24).

[WJC20] Changyun Wei, Ze Ji, and Boliang Cai, « Particle Swarm Optimization for Cooperative Multi-Robot Task Allocation: A Multi-Objective Approach », *in*: *IEEE Robotics and Automation Letters* 5.*2* (Apr. 2020), pp. 2530–2537 (cit. on p. 28).

[WS16] Zijian Wang and Mac Schwager, « Kinematic multi-robot manipulation with no communication using force feedback », *in*: *2016 IEEE International Conference on Robotics and Automation (ICRA)*, Stockholm, Sweden: IEEE, May 2016, pp. 427–432 (cit. on pp. 11, 24).

[Wu+19] Shuang Wu et al., « Leader-following Consensus and Trajectory Tracking for Nonholonomic Mobile Robots », *in*: *Proceedings 2018 Chinese Automation Congress, CAC 2018*, IEEE, 2019, pp. 3678–3683 (cit. on p. 32).

[WVB19] Fang Wu, Vivek Shankar Varadharajan, and Giovanni Beltrame, « Collision-aware Task Assignment for Multi-Robot Systems », *in*: *2019 International Symposium on Multi-Robot and Multi-Agent Systems (MRS)*, IEEE, Aug. 2019, pp. 30–36, arXiv: `1904.04374` (cit. on p. 74).

[Xue+21] Kai Xue et al., « An Exact Algorithm for Task Allocation of Multiple Unmanned Surface Vehicles with Minimum Task Time », *in*: *Journal of Marine Science and Engineering* 9.*8* (2021), p. 907 (cit. on p. 27).

[Yan+04] Xin Yang et al., « A decentralized control system for cooperative transportation by multiple non-holonomic mobile robots », *in*: *International Journal of Control* 77 (Aug. 2004), pp. 949–963 (cit. on p. 24).

[YJC13] Zhi Yan, Nicolas Jouandeau, and Arab Ali Cherif, « A Survey and Analysis of Multi-Robot Coordination », *in*: *International Journal of Advanced Robotic Systems* 10.*12* (Dec. 2013), p. 399 (cit. on pp. 16, 17).

[YN08] Chika Yoshioka and Toru Namerikawa, « Formation Control of Nonholonomic Multi-Vehicle Systems based on Virtual Structure », *in*: *The International Federation of Automatic Control* 41.*2* (July 2008), pp. 5149–5154 (cit. on p. 20).

[Zel+15] Daniel Zelazo et al., « Decentralized rigidity maintenance control with range measurements for multi-robot systems », *in*: *The International Journal of Robotics Research* 34.*1* (Jan. 2015), pp. 105–128 (cit. on p. 37).

Titre : Navigation coopérative d'une flotte de robots mobiles

Mot clés : Systèmes multi-robots, contrôle de formation, algorithme de consensus, optimisation quadratique hiérarchique, allocation de tâches basée sur l'optimisation et les enchères.

Résumé : L'intérêt pour l'intégration des systèmes multi-robots (MRS) dans les applications du monde réel augmente de plus en plus, notamment pour l'exécution de tâches complexes. Pour les tâches de transport de charges, différentes stratégies de manutention de charges ont été proposées telles que : la poussée seule, la mise en cage et la préhension. Dans cette thèse, nous souhaitons utiliser une stratégie de manipulation simple : placer l'objet à transporter au sommet d'un groupe de robots mobiles. Ainsi, cela nécessite un contrôle de formation rigide. Nous proposons deux algorithmes de formation. L'algorithme de consensus est l'un d'entre eux. Nous adaptons un contrôleur de flocking dynamique pour qu'il soit utilisé dans le système à un seul intégrateur, et nous proposons un système d'évitement d'obstacles qui peut empêcher le fractionnement tout en évitant les obstacles. Le deuxième contrôle de formation est basé sur l'optimisation quadratique hiérarchique (HQP). Le problème est décomposé en plusieurs objectifs de tâches : formation, navigation, évitement d'obstacles et limites de vitesse. Ces tâches sont représentées par des contraintes d'égalité et d'inégalité avec différents niveaux de priorité, qui sont résolues séquentiellement par le HQP. Enfin, une étude sur les algorithmes d'allocation des tâches (Contract Net Protocol et Tabu Search) est menée afin de déterminer une solution appropriée pour l'allocation des tâches dans l'environnement industriel.

Title: Cooperative navigation of a fleet of mobile robots

Keywords: Multi-robot system, formation control, consensus algorithm, hierarchical quadratic programming, optimation-based task allocation, auction-based task allocation.

Abstract: The interest in integrating multi-robot systems (MRS) into real-world applications is increasing more and more, especially for performing complex tasks. For load-carrying tasks, various load-handling strategies have been proposed such as: pushing-only, caging, and grasping. In this thesis, we aim to use a simple handling strategy: placing the carrying object on top of a group of wheeled mobile robots. Thus, it requires a rigid formation control. A consensus algorithm is one of the two formation controllers we apply to the system. We adapt a dynamic flocking controller to be used in the single-integrator system, and we propose an obstacle avoidance that can prevent splitting while evading the obstacles. The second formation control is based on hierarchical quadratic programming (HQP). The problem is decomposed into multiple task objectives: formation, navigation, obstacle avoidance, velocity limits. These tasks are represented by equality and inequality constraints with different levels of priority, which are solved sequentially by the HQP. Lastly, a study on task allocation algorithms (Contract Net Protocol and Tabu Search) is carried out in order to determine an appropriate solution for allocating tasks in the industrial environment.