6-1-2018

Cooperative transport in swarm robotics. Multi object transportation

Aalaa Ibrahim Hamouda

THE AMERICAN UNIVERSITY IN CAIRO

MASTER'S THESIS

Cooperative Transport in Swarm Robotics

MULTI OBJECT TRANSPORTATION

Author:
Aalaa I. Hamouda

Supervisors:
Dr. Khaled El-Ayat

A thesis submitted in fulfillment of the requirements
for the degree of Master of Science

in the
Department of Computer Science and Engineering

May 13, 2018

The American University in Cairo

Abstract

Department of Computer Science and Engineering

Master of Science

Cooperative Transport in Swarm Robotics
Multi Object Transportation

by Aalaa I. Hamouda

Swarm robotics is a research field inspired from the natural behavior of ants, bees or fish in their natural habitat. Each group display swarm behavior in different ways. For example, ants use pheromones to trace one another in order to find a nest, reach a food source or do any operation, while bees use dance moves to attract one another to the desired place. In swarm robotics, small robots attempt to mimic insect behavior. The robotic swarm group collaborate to perform a task and collectively solve a given problem. In the process, the robots use the sensors they are equipped with to move, communicate or avoid obstacles until they collectively do the desired functionality. In this thesis, we propose a modification to the Robotic Darwinian Particle Swarm Optimization (RDPSO) algorithm. In the RDPSO, robots deployed in a rescue operation, transport one object at a time to a desired safe place. In our algorithm, we simultaneously transport multiple objects to safety. We call our algorithm Multi Robotics Darwinian Particle Swarm Optimization (MRDPSO). Our algorithm is developed and implemented on a VREP simulator using ePuck robots as swarm members. We test our algorithm using two different environment sizes complete with obstacles. First implementation is for two simultaneous object transported but can be extended to more than two. We compare our new algorithm to the results of single RDPSO and found our algorithm to be 35 to 41 % faster. We also compared our results to those obtained from three selected papers that are Ghosh, Konar, and Janarthanan [1], TORABI [2], and Kube and Bonabeau [3]. The performance measures we compare to are the accuracy of transporting all objects to desired location, and the time efficiency of transporting all the objects in our new system.

Acknowledgements

I would like to express my sincere appreciation and thank my supervisor Dr. Khaled El-Ayat, for all the help he provided through out all the time I was working on my thesis under his supervision and for all the support and pushing for this work to come to existence ...

Next I would like to thank both my husband and my life partner Mohamed and my sister and my twin Israa for all the effort they provided in this work and all the sleepless nights they stayed helping and supporting me. Without them, I would have never finished this huge milestone in my life ...

I would also like to thank for sure my parents and brothers for everything they did. I would have never been at this point of my life without their help, support and pushing forward, I would like to provide more thanks to my mother who was, is and will always be my superhero in life ...

Another big thank you needs to be addressed to my amazing best friend, Noha, thank you for everything you did, whether helping through the thesis or with the support you provided starting day one with your huge believe in me that I can make it and you were right, I did it ...

Finally, I would like to thank all my big family members, especially my cousins Sondos, Sarah and Sama for everything they provided. Without your moral support I would have never made it ...

Contents

List of Figures

List of Tables

List of Abbreviations

VREP	**V**irtual **R**obotics **E**xperimentation **P**latform
ARGoS	**A**utonomous **R**obots **G**o **S**warming
RDPSO	**R**obotic **D**arwanian **P**article **S**warm **O**ptimization
EPSO	**E**xtended **P**article **S**warm **O**ptimization
PEPSO	**P**hysically **E**mbedded **P**article **S**warm **O**ptimization
GSO	**G**lowworm **S**warm **O**ptimization
Multi-Objective PSO	Multi-Objective **P**article **S**warm **O**ptimization
MRDPSO	**M**ulti **R**obotic **D**arwanian **P**article **S**warm **O**ptimization
PFSM	**P**robabilistic **F**inite **S**tate **M**achine
MOPSO	**M**ulti **O**bjective **P**article **S**warm **O**ptimization
NSGA-II	**N**on-Dominant **S**orting **G**enetic **A**lgorithm-II

Dedicated to my husband and my sister for their effort that made this happen…

Chapter 1

Introduction

Swarm robotics is a field of robotics in which multiple robots are coordinated in a decentralized and distributed manner as defined by Navarro and Matía [4]. Swarm robotics takes inspiration from the social activities of insects in nature like ants, bees, .. etc. In nature, biological lifeforms can demonstrate this swarming behavior in a different ways to meet their needs. Take for example the ant, which uses pheromones to mark a path for other members of their colony to follow to a food source, or back to the nest. The highlighted path will continue to get stronger and stronger as each ant in turn will continue to mark the path. Once the path is no longer needed, the ants simply stop marking it, and the pheromones simply vanish along with the path. Bees, on the other hand, use dancing to attract the attention of other bees, drawing them to the source of food. The dancing, known as a waggle dance , serves as a way to of calling and recruiting other bees to help secure the food source. Swarm robotics borrows the concepts found in these biological behaviors, replacing insects with small robots.

In a swarm robotics system, the environment consists of an area where the robots explore, obstacles to navigate around, and a target. The aim of swarm robotics is to solve a problem collectively by communicating effectively about the environment in which the robots find themselves in. The task can be as simple as exploring the available area or as complex as interacting with or avoiding the objects in the environment with the goal of solving a problem. There are many different applications for swarm robotics, from search and rescue and military application, mining and gas exploration to modern medicine with advancements in nanotechnology as discussed by Hasan [5].

One application of particular importance and the focus of this thesis is search and rescue. During search and rescue operations there may be certain scenarios where the risk of sending rescuers into an unstable environment may be too dangerous and carry the potential of further loss of life. In this circumstances, special search and rescue robots could be deployed that are governed by swarm robotics. The robots would work collectively navigating around the environment, avoiding impediments to locate survivors. Once a survivor is located, the locating robot would

send a recruiting signal to other robots, and then when the number of robots is sufficient for the transportation task, the transporting process would begin to move the survivor to safety. This would be a valuable option for search and rescue teams to have at their disposal in times of need.

In this thesis, we will tackle the area of transporting multi objects at a time in the field of swarm robotics. The focus of this area is for the robots to transport multiple objects at the same time using a collective of similar robots which will perform the task in unison. In similar research studies, there are multiple techniques proposed and implemented for transporting an object from its current location to a destination defined by the application. Transporting more than one object at a time will result in a reduction in the time needed to finish the given task as long as it can be accomplished without a risk to the efficiency of transporting the objects themselves.

Swarm robotics has the potential to provide many advantages over other similar systems to solve problems. These advantages, summarized below, will be discussed in more detail throughout this thesis. They are as follows:

- Parallelizable: the ability for multiple robots to perform tasks in parallel in order to reach a specific goal.
- Scalable: the decentralized and distributed nature of swarm robotics allows for the entire system to scale as needed. The addition or removal of any number of robots does not harm the overall system.
- Robustness: no single point of failure, because the tasks are shared across all robots, the loss of one or more robots does not affect the system.
- Reusable: the presence of multiple small functions in the system that can be reused to perform bigger tasks in the system.
- Economical: the use of simple robots in a swarm system provides significant cost savings and fault tolerance when compared against larger more complex robotics systems.

The main contributer of our thesis is in two parts, the first part in the recruitment technique and the second part in transporting multiple objects at the same time. As for the first part, the recruitment process that we succeeded in implementing recruits two group of robots for two different targets. In this process, each group is only concerned with the target they are recruited to transport. As for the second part, the multi object transportation that we succeeded to conduct was successful due to the success of the recruitment process and that every target has its own path with no overlap between the two targets.

1.1 Problem Definition

The primary focus of our thesis is in the area of cooperative transportation with the aid of swarm robotics. Cooperative Transportation, a relatively new field, is simply defined by Czaczkes and Ratnieks [6] as "multiple individuals simultaneously moving an object". When applied to swarm robotics the idea is that a group of robots, either similar or varied in architecture, work collectively to transport a specific object from one place to another, all while performing various tasks like judging distance and speed over time, and object avoidance. While researching similar projects, which will be discussed in detail in Chapter 2, various ways of transporting an object were found. These ways included pushing or pulling from either the front of the object or its back. This would be done by having the recruited forming a circle around the object and then once in position begin the process of transporting the object.

Communication between the robots is essential for the transportation process to be effective. The most effective way of communication found, and the one used in this thesis, is where one the discovering robot would send out recruitment signals which would include the location of the object and all responding robots would be recruited to assist in the transport process. Another method would involve the recruiting robot to travel back to the home of the robots and start the recruiting process from there. Many of the other projects researched only dealt with single object transportation, and did not address the challenges of multi object transportation, leaving the area open to new discovery.

In nature, insect colonies, like ants and bees, have long since perfected the art of cooperative transportation and one would be more likely to observe instances of these insects moving more than one object at time instead of moving a single object. The coordination between these subject is highly complex and allows them to speed the process of food collection, nest building or any other process required by the colony. So, since the whole robotics swarm was inspired from nature swarm, and the transporting of only one object has already been explored and multiple techniques that help in performing the transportation arose, we decided to explore the transportation of multiple objects at a time, and hence, starting the exploration with the aid of more than one object to be transported at a time.

There are many challenges to multi-object transportation that this thesis addressed through its development and will be more illustrated in 5. Many of these challenges still pose complex problems for single-object transportation, for example, robot coordination to move or transport an object. Of course there are some new challenges that will arise due to introduction of adding multiple transport objects that need to be transported at the same time. The communication within 2 or more teams

of robots transporting multiple objects in unison. Each robot on the separate team must only communicate with other robots of the same team. Any communication with robots outside of the team would cause issues with navigation, object avoidance, and/or the ability of the robot to gauge how much to push or pull the object. The robots must see only the object that they will transport and all other objects (including other robots) as obstacles to avoid otherwise the confusion could cause a failure across the entire team of robots.

Some of the other challenges beyond that which are found in the transportation case are to do with robot movement, multi-robot coordination, order of object transportation, robot recruitment, choosing the correct recruitment technique, synchronization between robots while transporting the object to ensure the object reaches its destination accurately, differentiating different objects to be transported and once an object is detected, the robot must start the recruitment phase and transportation phase without checking other near objects, each robot to be able to differentiate which group of robots it belongs to, to avoid any confusion while recruiting and transporting, and of course, obstacles avoidance to not hit any other objects or obstacles scattered in the environment. Again all these challengers are conquered along the way. These challenges as well as others will be addressed in this thesis.

The primary problem discussed in this thesis will focus on simultaneous multiple objects transportation. As discussed earlier, all swarm species in nature can handle multi-object transportation. In our thesis, we will apply simultaneous multiple objects transportation on a well known problem in swarm systems, search and rescue application. Search and rescue applications are concerned with saving victims. A group of robots can be deployed in an emergency situation to help search and rescue victims, robots are the best candidate for such application as losing robots unlike losing people can be acceptable in these situations. Another application for transporting multiple objects at a time is arranging warehouse products, again by scattering multiple robots to locate the objects/products to be stored and transport them to the storage places. We proposed a solution for this problem in Chapter 3.

The thesis document is divided as follows. Chapter 2 provides a literature review explaining all approaches found in the literature regarding both swarm robotics and cooperative transport. Our proposed solution along with the experimental environment to be used in this thesis are described clearly in Chapter 3. Finally, Chapter 4 concludes our thesis and presents our future work. As for our platform determination, Appendix A includes all the needed details regarding our comparison to support our choices for the platforms. Appendix B contains our implemented algorithm brief description along with its pseudo code.

Chapter 2

Literature Review

2.1 Swarm Robotics

Swarm robotics is inspired from nature as swarm in general represents a group: swarm of ants, bees, fish ... etc as seen in Figure 2.1 adopted from Tan and Zheng [7]. So, swarm robotics is based on how a swarm of robots behaves leading to new potentials. Swarm in natural can be seen in ants; how ants behave and communicate together using pheromones to reach food source; in bees; how bees use dancing techniques to get attention of other bees to a potential food source in the area; in fish; how fish communicate together and form a shape; in birds; how birds communicate and start migration from one place to another. Swarm robotics coordinates the behavior of a group of robots to perform a specific task taking into consideration some complications depending on the environment. This is achieved through controlling both the communication and the interaction between robots.

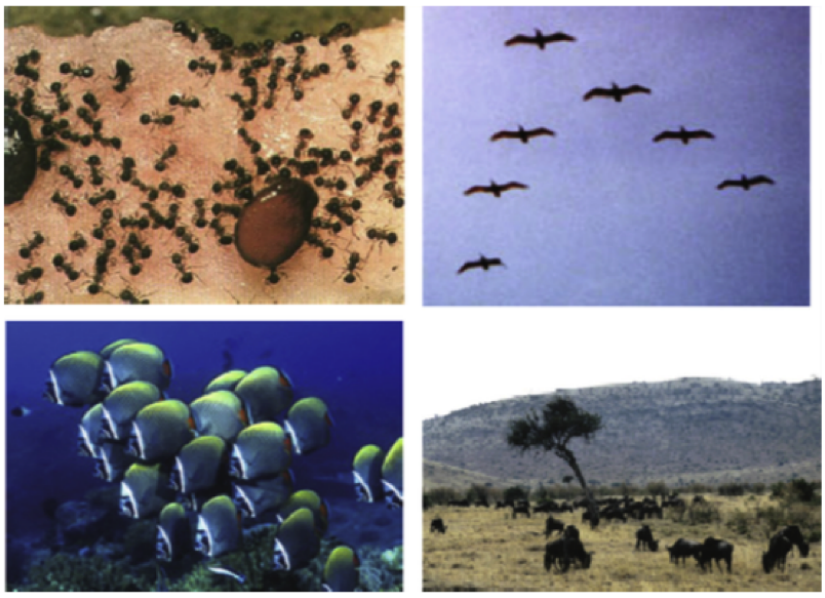

FIGURE 2.1: Biological swarm in nature adopted from Tan and Zheng [7]

Ant colonies are a good example of swarm as the ants start the searching process for

food by exploring the environment around them; they start moving randomly creating new paths along the way using their pheromones. Only the paths that lead to food source gets more and more pheromones due to ants using them over and over again leaving more and more pheromones traces behind them. As for the rest of the paths that lead to nothing no more ants are going through them so the pheromones traces keep fading away until these paths no longer exist.

Another good example of swarm is the bees as they start also by exploring the environment around them searching for food. When a patch of flowers is found, the bee starts dancing as shown in Figure 2.2 adopted from Seeley, Visscher, and Passino [8] to let other bees know that it found the flowers and so other bees start approaching the patch of flower. The bees use this dance to communicate with other bees the distance, and direction of the patches of flowers, the water sources or the new hive locations. The bee dance is known as the waggle dance, the direction the bee dances represents the movement respectively with the hive and the distance from the hive to the found source is represented by the duration of the dance.

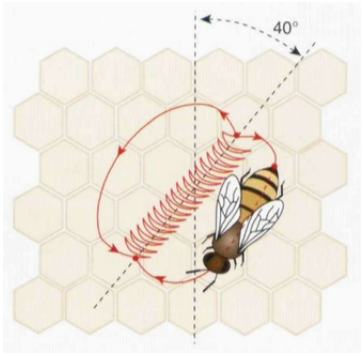

FIGURE 2.2: Bee Dance adopted from Seeley, Visscher, and Passino [8].

Humans also apply the swarm techniques in a form known as teamwork, where a group of people works together to perform a specific task. For an example, consider a software company with a website as a project, it all starts with a group of software developers working together to launch the website, they start dividing tasks around them where some smaller teams are being performed, each team is assigned part of the tasks to deliver the whole project, a group takes the frontend part, another group works on the backend part, another group handles the user interface, group for the documentation and so on.

Swarm robotics can be achieved through multiple techniques. Either all robots have the same functionality. This means that any robot in the swarm is capable of performing any of the small tasks to reach the goal task. In this technique, if any point in the system, which is represented by a robot fails, still the big goal will be reached, as any other robot will perform its task. Another technique is to divide the big swarm

of robots to smaller groups where each group has the functionality of one small task of the big goal. This means that each group depends on the other and if there was a failure in one group the task will not be performed. Hence, the big goal will not be reached. But this is easily overcome by placing multiple robots in the group, i.e. no group consists of only one robot.

2.1.1 Swarm Robotics Applications

Swarm robotics is being explored since years now, during these years multiple problems or can be called applications have been solved with the aid of swarm robotics, from these problems/applications are aggregation, pattern formation, self assembly, chain formation, foraging, dispersion, coordinated movement, collective exploration, navigation and cooperative transport. In this section, we will discuss briefly each application.

Foraging

Foraging, by humans or animals, is the act of searching for and collecting food for storage or consumption. Robot foraging is defined more broadly as searching for and collecting any objects, then returning those objects to a collection point or target location.

In robotics, foraging is the process of searching for the nearby objects and return to a specific goal place with the maximum number of objects. It is inspired from ants searching around the nest to maximize the amount of food returned.

Foraging is important for several reasons: firstly, it is a metaphor for a broad class of problems integrating exploration, navigation and object identification, manipulation and transport; secondly, in multi-robot systems foraging is a recognized problem for the study of robot-robot cooperation, and thirdly, many actual or potential real-world applications for robotics are instances of foraging robots, for instance cleaning, harvesting, search and rescue, land-mine clearance or planetary exploration.

Aggregation

The attempt to collect all the robots together, or make them come near one another is known as aggregation as shown in Figure 2.3 in Trianni, Groß, Labella, *et al.* [9]. This is an important problem as it is considered the first step needed in other problems, since collecting the robots together to perform a task is the aim of swarm robotics. The main two approaches in dealing with the aggregation problem are probabilistic

finite state machine (PFSM) and evolutionary algorithms.

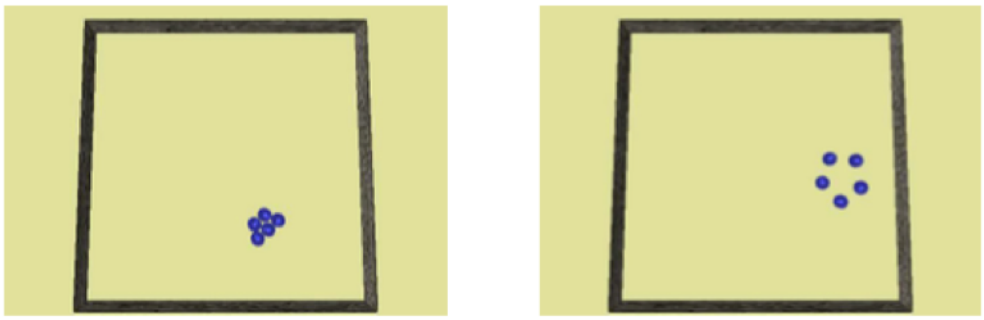

FIGURE 2.3: Static and dynamic aggregation example.

Pattern Formation

This problem is related to forcing the robots to collectively form a specific shape while preserving a specific distance between one another as shown in Figure 2.4 in Brambilla, Ferrante, Birattari, *et al.* [10]. The shape is determined before starting the experiment and depending on some local information regarding the robots' positions. The main approach in dealing with such problem is virtual physics based design.

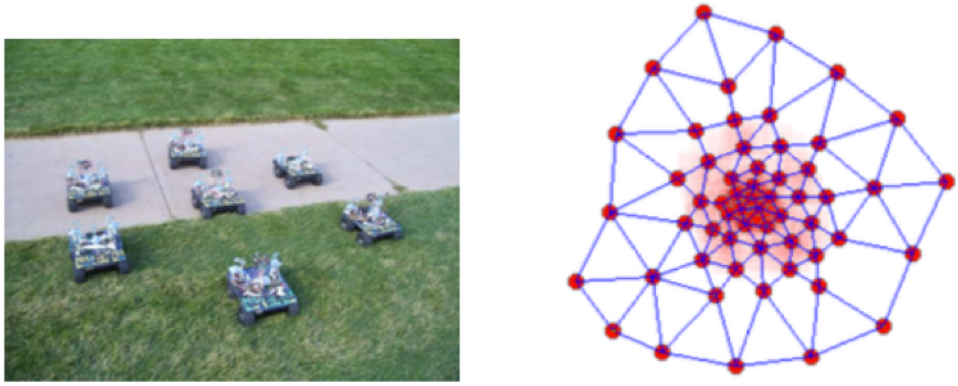

FIGURE 2.4: Pattern formation example.

Self Assembly

In this problem robots tend to self-assemble meaning attach to one another to form a specific pattern that will make solving the issue easier as shown in Figure 2.5 adopted from Brambilla, Ferrante, Birattari, *et al.* [10] . As an example, robots get attach to one another in the form of a straight line to pass a hole on their way. This problem is usually divided into two parts one related to robots assembling, which is called morphogenesis, and the second part is related to controlling the new-formed

shape. The main approach to tackle this problem with its two parts is probabilistic finite state machine (PFSM), while the second part of the problem can be also tackled using artificial evolution.

FIGURE 2.5: Self assembly example.

Chain Formation

The robots communicate together to form a chain as shown in Figure 2.6 adopted from Brambilla, Ferrante, Birattari, *et al.* [10] that can be used after that as an identification path to the object or can be used in surveillance. The three main approaches for this are probabilistic finite state machine (PFSM), virtual physics based design and artificial evolution.

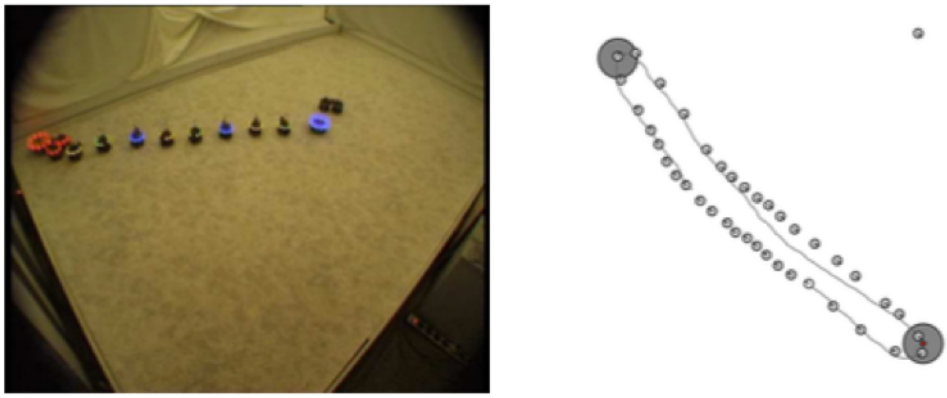

FIGURE 2.6: Chain formation example.

Dispersion

This is the other side of aggregation. In this problem, the robots spread out in the environment to cover the maximum area possible and still maintaining some sort of communication between one another. This is used in surveillance. Tackling this problem can be performed using either probabilistic finite state machine or artificial

evolution.

Coordinated Movement

Movement of a group of robots to conduct a specific task, while maintaining robots' communication, no collision between robots and no collision with obstacles in the environment is shown in Figure 2.3 adopted from Brambilla, Ferrante, Birattari, *et al.* [10]. This problem is considered the basic block for many other problems, as robots' movement is the starting point of performing any task related to swarm. The approaches available to deal with such problem are virtual physics based design and artificial evolution.

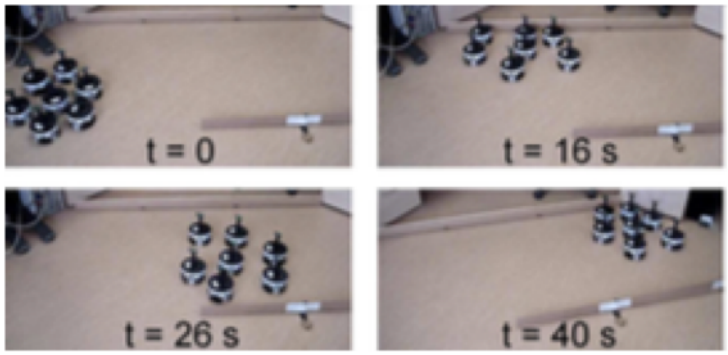

FIGURE 2.7: Coordinated movement example.

Collective Exploration

Spreading the robots all over the environment in order to search for something or to locate something is what is known as collective exploration as shown in 2.8 adopted from Brambilla, Ferrante, Birattari, *et al.* [10]. Robots still maintain communication between one another to communicate back the object in need position, also robots move seamlessly in the environment without colliding with each other or with any obstacle. The approach in literature that addresses such problem is virtual physics based design.

Navigation

After the robots are done with searching or exploring the environment, one task they can do is construct a path to an object. An example is bomb diffusion problem. In this problem, bombs are scattered in the environment, robots start by searching for the bombs, and once a robot detects a bomb it communicates its location to the

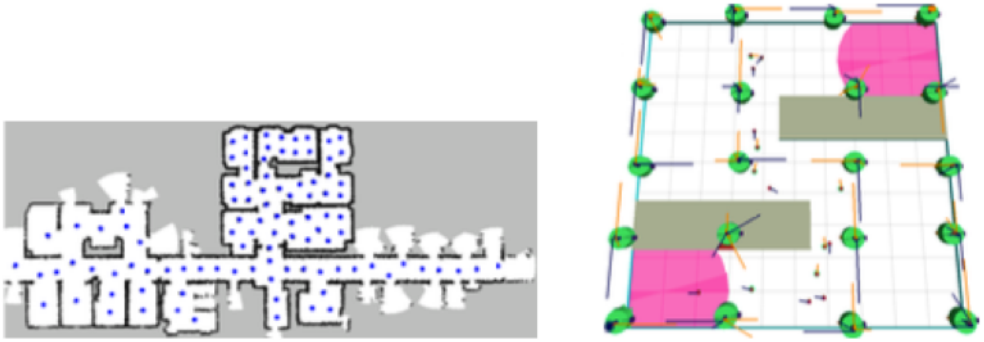

FIGURE 2.8: Collective exploration example.

nearby robots constructing a path through robots from the bomb position to the human.

Collective Transport

This problem is the one that is going to be tackled in this thesis. Cooperative transport is the problem of moving an object from one place to another with the help of a swarm of robots as shown in Figure 2.9 adopted from Brambilla, Ferrante, Birattari, *et al.* [10, postnote]. This approach is dealt with in literature using probabilistic finite state machine (PFSM) or artificial evolution. Cooperative transport is going to be discussed in details in the next point.

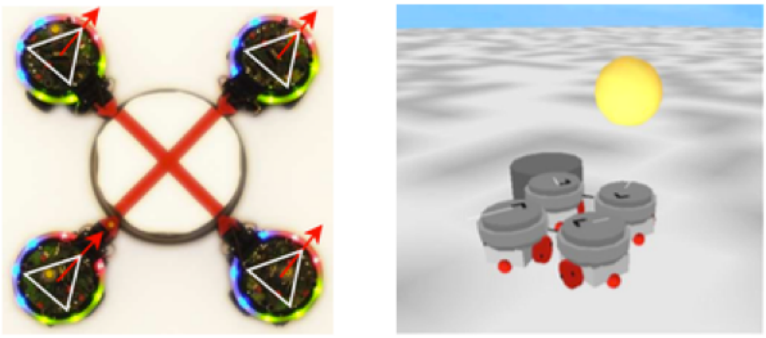

FIGURE 2.9: Collective transport example.

2.1.2 Swarm Robotics Challenges

Constructing swarm robotics systems is a very challenging task, as this system requires taking into consideration multiple criteria. One of the major challenges is maintaining communication between robots. Robots should always sustain a communication channel to keep exchanging information about robots' status, obstacles' positions and all other information regarding the surrounding environment. Another challenge is concerned with the robot movements; there are difference types

of movements regarding robots. The starting movement where each robot moves without taking into consideration any movement from other surrounding robots, and this can be called the exploration movement. The movement when the target is found, during this movement the robot tries to collect nearby robots to perform the task needed together, this can be called the searching movement. The movement where all robots together do the assigned task and this movement can be called coordinated movement. The mentioned challenges are crucial as they represent the fundamental stages in constructing the system.

The relationship between the robots is another challenge that should be considered. This relationship can be either homogeneous or heterogeneous. Homogeneous systems are the systems in which all robots have the same functionality installed. They all do the same task. All robots are duplicated from one another. While heterogeneous systems are the systems in which robots are subdivided into two or more groups, each group is assigned a different task than the other groups but collectively all groups perform the final task. Selecting the best type of relationship depends on the task to be performed, the number of available robots, and the surrounding environment.

Another thing to be taken into consideration is the type of algorithm to be used to perform the task. The two main categories under which the algorithms lay are centralized algorithms and decentralized algorithms. Centralized algorithm is where there is a main station that transmits and receives from the robots. This station is responsible for all type of communication between the robots, saves all the received information from the robots regarding their positions, obstacles and so on. Also, exchange between robots if one of them reached the goal, or if one of them needs others to help with performing the task and so on.

As for the decentralized algorithm, in this algorithm each robot can communicate with other robots without the need for a station. Each robot stores the information regarding its position and receives information regarding the robots around it. Each robot can signal the other surrounding robots if it needs help performing the required task and is capable of transmitting obstacles' positions to nearby robots that retransmit it to other robots. At the end, in the decentralized algorithm, each robot or group of robots is a complete entity that can perform all tasks. Selection between centralized and decentralized algorithms is also dependent on the type of task to be performed, the resources available, the number of robots and the surrounding environment.

Another challenge ahead is the configuration of the robots, this configuration is the robots' way to perform a task that can't be performed by only one robot. This configuration can either be self-assembly or self-reconfigurable. Self-assembly is where

a group of robots perform a task in which each robot can't do it alone, so they for example attach to one another forming one big robot that can perform the task, then they get detached in to individual robots again to continue remaining functionalities. On the other hand, self-reconfigurable is where a group of robots can attach to one another forming different configurations depending on what is needed or depending on the functionality. For an example, robots need to go from one place in the environment to another place separated by a ramp, they form a straight line to help push one another to that place, if the functionality is to transport an object so they form a circle around the object to transport it and so on. After concluding the task, they detach from one another and continue other tasks in the system.

2.1.3 Swarm Robotics Characteristics

All these advantages make swarm robotic systems highly considered in solving problems. After discussing the previous aspects of swarm robotics and knowing the challenges and advantages of such systems. It's time now to talk about the characteristics that make these systems so appealing to solve problems. These characteristics are autonomous, decentralization, homogenous, flexible, sensing and communication, and simple and are discussed in Tan and Zheng [7],Sahin, Girgin, Bayindir, *et al.* [11],Brambilla, Ferrante, Birattari, *et al.* [10],Mohan and Ponnambalam [12],Navarro and Matía [4],Barca and Sekercioglu [13], and McCreery and Breed [14].

Autonomous is one property of the robots included in the system. They don't depend on one another to perform any task. Any robot in any position in the system will be able to function and finish the task assigned. This can be understood by recalling the example of robots mentioned in re-usability advantage. Each robot does not know what other robots around do, it just functions according to the situation. If it was able to locate an object then its function now is to propagate to others that the object is found and communicate the object's location, if it got the message of an object available along with where it is located and the robot is near to the communicated position so it will head to the object to help other near robots in transporting the object, if the path to transport the object is not clear yet and the robot in searching state while other robots are moving the object then the robot might start looking for the best path to reach target position and send it to other robots.

Decentralization was discussed in terms of algorithms while talking about the challenges in swarm robotics and the difference between it and centralized algorithms. Robots can benefit from this property to speed up finishing the task, as the robots don't need to wait for a central control to communicate the action to be done, as this action is already defined within the logic of the robot as mentioned multiple times in this thesis, each robot performs according to its position in the environment or

the system, it decides the needed functionality at this point and act upon it without returning to a centralized control system. Decentralization helps maintain the scalability. Going back to scalability it is the act of adding or removing a robot to or from the system without causing any malfunction in the system, so it is obvious that decentralization helps with it.

Homogeneous was also explained with its complement algorithm heterogeneous. Homogeneous in swarm is due to all robots being the same. All robots have the same capabilities; they all have the same tasks assigned and they choose the task needed according to what is faced in the environment. Homogeneous is important as it maintains the robustness and the usability of swarm systems. Recalling robustness and usability, they both depend on robots being exact copy. At the end of the day any robot can perform all the functionalities required by the system and any robot can replace any faulty robot in the environment.

Flexibility in swarm is due to that multiple small robots that can perform any task in the system and in the same time with a little more hardware and with some tweaking to the software code, they can perform another task. Also, robots must have the flexibility to change a bit according to what they face in the environment. If we can recall the example in self-reconfigurable robots, according to the environment and the needed task the robots choose either the straight-line configuration or the circle configuration. This is one way for flexibility in swarm. Another form of flexibility is the one thing that was stated a lot through this thesis, which is the ability of each robot to perform in any part of the environment and their ability to be exchanged with one another without any disrupt to the system.

Local sensing and communication in swarm robotics system is one of the main problems in swarm due to having multiple robots and maintaining the communication all the time for the system to be connected. This can be done in multiple ways, either maintaining periodic communication by robots broadcasting their position and some information regarding the environment every now and then, or using some central station that adds a layer of communication between robots in broadcasting information that makes sure robots are not overwhelmed with data, also there is a way of making some robots like leaders to robots, these robots receives the information all the time and passes this information to the group periodically and one more way is robots always keep communicating with nearby robots only and by this the whole environment is connected.

Simplicity of swarm systems comes from the fact that they are all similar robots, all have the same functionalities, all have all tasks coded and they choose according to surrounding environment, failure of any robot does not affect the task or the system and addition of robots too is acceptable. Being that simple makes swarm systems a

good choice for big problems. Simplicity of swarm also arises from the fact that it takes big problems and divides it into multiple simple smaller ones. Solving multiple simple divided tasks make the system maintains its simplicity and makes the big problem easier to handle and solve.

2.1.4 Swarm Robotics verses Similar Systems

In this section, we will discuss all available systems similar to swarm robotics systems, these systems are multi-robot systems, multi-agents systems and sensor network systems. These systems are alike the swarm systems and can be mistaken to. Also, they are always put in comparison with swarm systems and sometimes in confusion when deciding which system to choose as they all are inspired from the natural swarm and they all try to make use of the cooperative nature of swarm systems.

Multi-robot systems as illustrated in Table 2.1 are systems with small robots population that depend in robot control on centralized or remote algorithms. The system is known to be usually homogeneous with an environment that can be either known or unknown. The robots in such systems have the ability to move in the environment. The disadvantages of such system are it has low flexibility and low scalability.

Sensor network systems as explained in Table 2.1 are characterized by fixed population of robots, also depend on centralized or remote algorithms when dealing with robot control. These systems are also homogeneous in their nature but the robots don't have the ability to move and explore the environment, as the environment for such systems is a known environment. These systems have low flexibility but moderate scalability.

Multi-agent systems; as described in Table 2.1; are characterized by small population of robots. When it comes to robot control they depend on either centralized or hierarchical or network algorithms. The environment is known to be either homogeneous or heterogeneous with rare ability for the robots to move around in the environment, the environment in this system is a known environment. As for the flexibility and scalability of these systems they are moderate in both.

As for swarm robotics systems and as obvious in Table 2.1, they are characterized with great population of robots, depending in robot control on decentralized and autonomous control algorithms as mentioned before in the characteristics of swarm systems. These systems are homogenous system in unknown environment with the full movement ability of the robot to explore and interact with the environment. Finally, from their advantages is they are highly flexible and highly scalable which was mentioned before along with the other mentioned advantages.

TABLE 2.1: Swarm robotics systems versus similar systems.

Criteria	Swarm Robotics	Multi-Robot Systems	Sensor Network Systems	Multi-Agent Systems
Population Size	Great Range	Small	Fixed	Small Range
Control	Decentralized and autonomous	Centralized or remote	Centralized or remote	Centralized or hierarchical or network
Homogeneity	Homogeneous	Usually homogeneous	Homogeneous	Homogeneous or heterogeneous
Flexibility	High	Low	Low	Medium
Scalability	High	Low	Medium	Medium
Environment	Unknown	Known or unknown	Known	Known
Motion	Yes	Yes	No	Rare
Typical Applications	Post disaster relief Military applications Dangerous applications	Transportation Sensing Robot football	Surveillance Medical care Environmental protection	Net resources management Distributed control

2.1.5 Swarm Robotics Advantages

The above-mentioned challenges are all important to be covered to help reach a system of swarm robotics that is reliable and perform in a good way. This takes us to a very important part, which is the advantages of swarm robotics; the reasons people go through all the mentioned challenges and more to use such systems. These advantages are parallelism, scalability, stability, robustness, reusable, being economical, and finally energy efficient and are discussed in Tan and Zheng [7],Sahin, Girgin, Bayindir, *et al.* [11],Brambilla, Ferrante, Birattari, *et al.* [10],Mohan and Ponnambalam [12],Navarro and Matía [4],Barca and Sekercioglu [13], and McCreery and Breed [14].

Parallelism is achieved as each swarm system contains large number of robots, by dividing the number of robots into small groups each containing couple of robots and each group is assigned one task in the system; dividing the big problem solved by the system into small tasks; at the end, each group will solve one task resulting in multiple tasks finished at the same time, hence, parallelism in finishing tasks. Another example of parallelism can be shown if the system is dealing with multiple objects, each group of robots can be assigned to deal with one object say for example transporting it, hence, parallelism in handling objects.

Scalability is the act of adding or removing a robot from or to any group resulting in robots performing coherently without any problems. Scalability is reached as all the

robots contain all the functionality in the system and they act according to the situation they are in. to give an example, assume the system is supposed to transport an object in an environment, functions within such system can be divided to first searching the environment to find the object and this can be done by one group, second searching for the best path to reach the target position and this will be finished by another group of robots and third pushing the object to transport it and this will be done by another group. When the system starts, no roles are assigned. No robot knows which group it will be in or what is the functionality it should do. So, deciding the task will be done according to searching and finding the object. Conclusion of this is any robot in the system can be in any group and will be able to perform any of the three mentioned tasks seamlessly.

Stability is easy to explain. The system is stable, no matter what happens it will perform in a stable manner. Adding robots, removing robots, adding functionalities, or removing functionalities, anything can happen and still the system will maintain its stability. This is very important as unstable systems won't be reliable and won't be considered in solving problems.

Robustness is the ability of the system to survive under any circumstances, as it is not going to be affected by any external forces. It is going to keep performing after adding some extra robots. It is not going to be affected by robots quitting. The functionalities still will be executed. If a system starts with a specific number of robots and as the system goes some robots start to malfunction due to any reason, the system still maintains its functionality with the remaining robots. Also, adding more robots won't affect the system as they will merge in the system and help the already available robots to perform the required tasks.

Reusability is accomplished due to the simple components of the swarm systems, also, due to that all these components are an exact copy from one another. As each robot can be reused for any functionality within the system since they are all programmed with the same code. To make this clearer, assume at the start of the system we have robot 1 and robot 2 searching in the environment, robot 3 and robot 4 were able to locate the object to be transported, robot 5 and robot 6 constructed the path to target position and this task was finished successfully. After that all robots are in searching phase, this time robot 2 and robot 5 located the object while robot 1 and 4 constructed the path to target position, leaving robot 3 and 6 in searching phase and so on. Any robot can be reused in any task within the system.

Economical is a very important advantage in swarm robotics. Economical property is represented in the cost of the project from one side and the other side is economical in finishing tasks. For the first side, constructing multiple small robots is much cheaper than constructing one huge one. Small robots consist of small components

that cost much less than components for a huge robot. As for the other side, swarm robotics is economical as having this huge population of small robots makes them perform the task more efficient than one single robot can.

Energy efficient is the last characteristic of swarm that will be discussed. Power is a critical aspect in any system not only swarms systems, as it determines how long the system will last. Energy in swarm systems is consumed in every movement and all the time, as the robots don't stop moving, even if they finished the task assigned they return to searching state, but since the robots are so simple, their energy consumption is not that high as the components used are simple small components. Most of the energy consumption will be in the communication, as this must be maintained all the time the system is alive, for all the robots to have knowledge of the surrounding environment and have knowledge of around robots. But still with this communication overhead, swarm robots will perform much better energy wise than one single robot executing the tasks.

2.1.6 Swarm Robotics Modeling Systems

Explaining what is there in literature regarding swarm robotics definition, challenges, advantages, characteristics and similar systems available, let's head to the next topic to be discussed in this thesis; which is swarm robotics modeling methods, modeling the swarm system to know will it be able to solve the problem in hand or not and with certain criteria as solving it only is not the purpose, it should be solved fast and with high accuracy. Modeling methods can be discussed in various points sensor based modeling, microscopic modeling and macroscopic modeling or there is the alternative method to modeling which is using real robots and getting the accurate results and compare them with the modeling results. Modeling of swarm systems is done through many simulators available nowadays.

Sensor based modeling defines the main components of the system as the sensors, the actuators and the objects in the environment, also defines the interactions between the robots and the interactions between the robots and the environment and are modeled as easy and simple as possible. This method is widely used to build more realistic swarm modeling systems, as the results of this method are in agreement with the results of experimenting on real physical robots. Two main factors should be taken into consideration while building such systems, simplicity and realism. The modeling of all interactions must be done in the simplest way possible to not alter the system speed. Also, realism factor should be done in way to maintain the results similar to those conducted from the real robots.

Microscopic modeling is conducted as sensor based modeling on the robot level. This model models the robots as entities, models the interaction between robot and

robot, models the interaction between robots and the environment and models every obstacle in the system. Everything is modeled as a separate entity. This way the whole system is modeled through the small entities and also through the interactions between the entities. The level of abstraction in microscopic modeling is very an important metric as it determines the accuracy of the retrieved results. There are three different levels of abstractions, the simplest level of abstraction, which reflects the robots as point-masses, adding more complexity leads to the second level of complexity, which assumes the environment as a 2D world with kinematic physics, adding extra complexity leads to the third and final level of abstraction which considers the environment as 3D world with dynamic physics. Microscopic modeling is what is used by the majority of the known simulators that help simulate swam robotic systems to help with obtaining the results.

Macroscopic modeling is conducted on swarm level not robot level. This method is the opposite of the microscopic method, where the whole swarm is modeled as one entity. The individual parts consisting the system such as the robots, the objects and the obstacles are all not considered for the aim of modeling the system at a higher level. This method of modeling gives the state of the whole swarm just by solving the system equation once, no need for iterations to get the results as needed by the microscopic method. All that is needed in this modeling technique is to find the system equation and solve it. There are multiple examples to macroscopic modeling equations in literature like using the rate equation, the differential equation, the partial differential equation, the langevin equation, the Fokker-plank equation and others.

Real robots as obvious from the name, just conduct the experiments on the robots as this will result in more accurate results than the modeling. There are a couple of difficulties in working with real robots as the fraction between the robots and the ground should be considered, this can't be shown in modeling except by adding some kind of fake fraction. Also, failure in communication can't be modeled through simulator; it is only a challenge when working with real robots. Most of the research available was conducted on simulators and there are multiple simulators out there to work with. Also, some chose real robots to experiment on which lead to the evolving of multiple platform for real robots.

2.2 Collective Transport

Swarm robotics as discussed in previous part is concerned with the use of a group of robots that can be identical or different, to perform a specific functionality that cannot be performed by one small robot and would be more efficient doing it with multiple small robots than using one big robot. In this thesis, cooperative transport

in swarm robotics is going to be discussed thoroughly, as it is the problem of concern that will be explained and will be tackled differently.

Cooperative transport is concerned with the deployment of a swarm of robots in known/unknown environment in order to transport one or more objects from one place to another. These robots can either be identical; all the robots know all the sub-tasks to be performed to reach the main goal, or they can be different; meaning the robots divided into sub-groups, each group has a specific sub-task to perform and collectively the groups perform the main goal. In cooperative transport, the main goal is of course transporting objects.

2.2.1 Ant Behavior

One of nature's species that inspired cooperative transport is ants. Ants although very small in size but collectively they can achieve big tasks. Ants are considered a very sophisticated system, as ants start by trying to find a nest for the colony. This nest must be big enough to fit the whole colony. The task of determining the best fit nest starts by ants exploring the surrounding environment trying to find any place that suitable to be the colony's nest then starts the second phase of taking other ants opinions in the nest, once this happens and multiple ants accept the new place as a suitable nest, starts the final phase which is moving to the nest. After settling, the ants start the surviving task of finding the food.

The task of finding food is the one of interest to this thesis, as this is the task concerned with transportation. Transporting different food sources in size and varies in distance to the nest. This task starts with ants spreading to try and cover as much as possible of the surrounding environment leaving behind traces of pheromones. Ants use these traces of pheromones to know their paths. If at the end of the path, a food source was found, then returning ants keep leaving extra traces of their pheromones to strengthen the traces on the food path, else if no source of food was found, the ants take the route back without adding any pheromones to the path. Other ants then check the different available paths according to traces of pheromones left by other ants and go with the path with the highest traces.

The task ahead for the ants is to transport the food source found to the nest. There can be two cases at this point in the task as described by Planqué, Van Den Berg, and Franks [15], either the food source is too small, hence, can be easily transported by one ant and this can be achieved by one of two methods as shown in Figure 2.10 adopted from Czaczkes and Ratnieks [6], either the object is so light that the ant lifts the object and move forward, or the object is heavy to be lifted so the ant starts dragging the object while moving backward, or the food source needs more than one ant to be transported and according to Figure 2.10 adopted from Czaczkes and Ratnieks

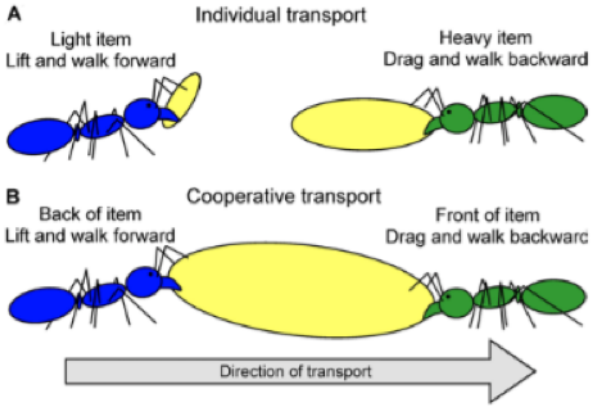

FIGURE 2.10: Individual vs cooperative transport in ants adopted from Czaczkes and Ratnieks [6].

[6], ants cooperate together through one ant lifting the back of the object and moving forward and the other ant dragging the front of the object and moving backward. In the latter case, the ant starts a recruitment phase. In this phase, recruitment can be achieved by different methods, tandem running in which an ant tries to lead another ant to the food source which in return leads another one and so on, group recruitment in which an ant can lead a group of ants of maximum thirty ants to the food source to help transporting it or scent trails. Deciding on the method of recruitment depends on the colony size.

After the recruitment phase comes the actual transportation. In Czaczkes and Ratnieks [6], cooperative transportation in ants in sub-divided into three categories, the uncoordinated transportation, the encircling coordinated transportation and the forward facing coordinated transportation. As for uncoordinated transportation, the transportation is assumed to be achieved through ants pulling the object to be transported, but ants do this motion in opposite directions to one another which leads to no forward motion due to the formation of multiple deadlocks. These deadlocks are solved through random behavior of ants, either random orientations, or random composition or random group behavior. Hence, uncoordinated behavior can be observed. Investigations in uncoordinated transportation suggest that transportation is done over three stages. Stage I, first ant locates the object to transport. Stage II, object gets located by multiple ants, deadlock occurs and transportation stops. Finally, stage III, deadlock is solved and transportation resumes. The final stage is characterized with high speed in motion and better construction of path.

In encircling coordinated transportation, the ants collectively encircle the object to be transported and start moving in coordinated motion to move the object. This category of transportation starts with one ant that tried to move the object, if the size of the object is bigger than the ant ability to move it other ants come to help forming a circle around the object. The ants at the back of the object lift the end part and move

FIGURE 2.11: Encircling coordinated transportation adopted from Czaczkes and Ratnieks [6].

forward, the ants at the front drag the front part and move backward and finally, the side ants either lift or drag the sides of the object and move sidewalks. All three motions should be coordinated between the ants to be able to move the object as shown in Figure 2.11 adopted from Czaczkes and Ratnieks [6]. Coordination is so obvious in this transportation category in the ants' movement.

In forward facing coordinated transportation, an ant finds a food source and starts immediately the transportation process by lifting the object from the front and begins moving forward. Along the way, other ants spots the struggling ant transporting the object and try to help by lifting either the sides or the end of the item and move forward as well. More ants can join at the back but at this stage they help by increasing the speed of transportation. this team of ants is shown in Figure 2.12 adopted from Czaczkes and Ratnieks [6].

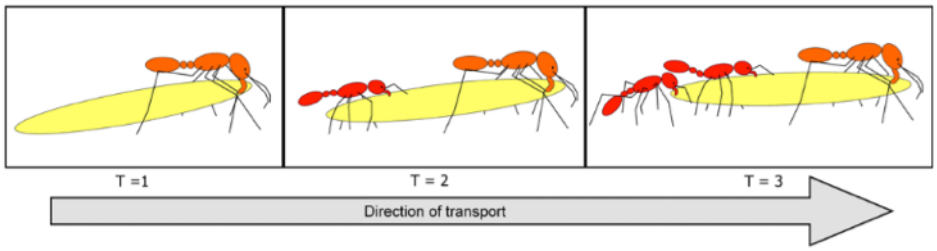

FIGURE 2.12: Forward facing cooperative transportation adopted from Czaczkes and Ratnieks [6].

2.2.2 Transportation Stages

Cooperative transportation in ants and also in swarm robotics is divided according to McCreery and Breed [14] in to four stages, the decision stage, the attraction/recruitment stage, the organization stage and finally, the transport stage. These four stages are shown as a flow chart in Figure 2.13 adopted from McCreery and Breed [14]. An ant spots a food source that is bigger than the ant's ability to move alone and here comes the decision stage as the ant needs to decide is this source worth being returned to the colony, if the answer is no, then it moves to search for

another food source else if the answer is yes then the ant takes the decision of recruiting other ants to help with food movement. Hence, it starts the recruitment phase. The recruitment is done by any method, the ant can return back to the nest to get other ants, or the ant can recruit neighboring ants around it or the ant can wait for other ants to spot the same source and move it with one another. Some of the ants will decide the movement directions, choose the path, and avoid obstacles and other organizational decisions or ants will start pulling/pushing in any directions without coordination and this is the third stage, the organization stage. At this time in the process, ants reach the final stage, which is the transportation stage where the ants are ready to move/transport the food source back to the nest.

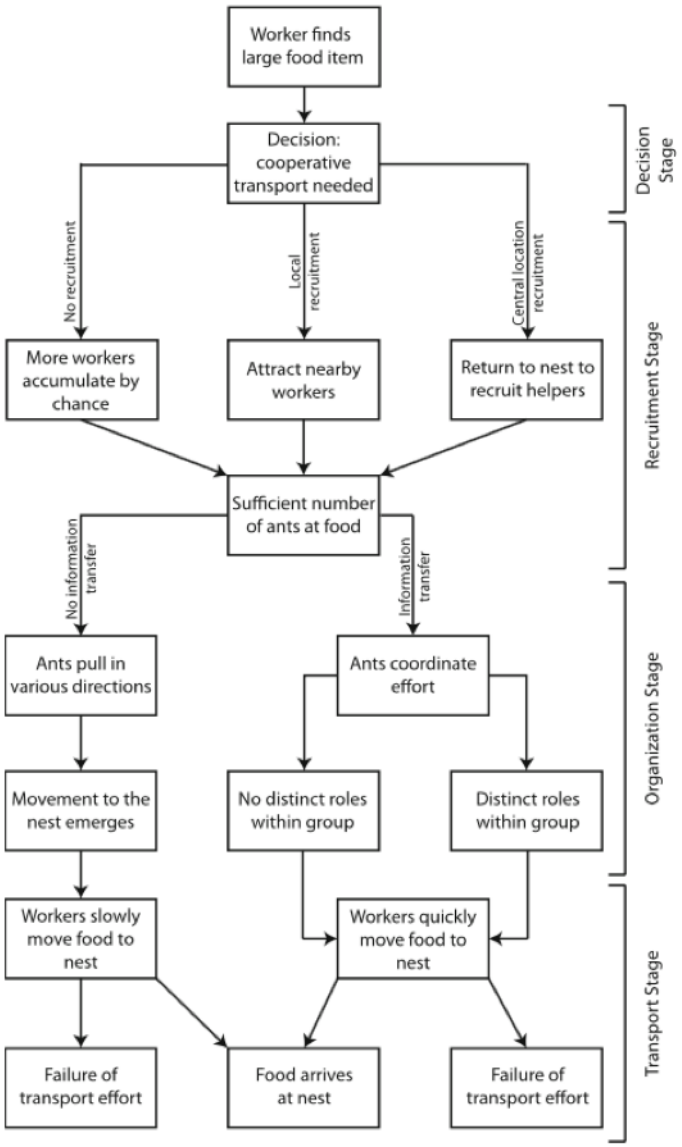

FIGURE 2.13: Cooperative transportation stage adopted from McCreery and Breed [14].

2.2.3 Transportation Techniques

Cooperative transportation in swarm robotics can be achieved through multiple techniques. The robots can use their accumulative force to push the object whether forward or backward to move it to the destination place. Also, the robots can use some attached grippers to hold the object and grip it or pull it to goal position. Another way is the robots form a circle around the object and coordinate their movement together toward the desired location. One extra way is that robots can be divided into four groups, two groups at the sides of the object helping with path alignment, one group at the back end of the object to push along the two groups at the sides and finally, the fourth group works as leader for the transportation process, helping in avoiding obstacles and making decisions regarding the best path to reach target.

2.2.4 Transportation Controllers

Many researchers have extensively explored cooperative transportation in the past years. Pereira, Campos, and Kumar [16] used caging technique to surround the object to be transported, controlling the system is done using a decentralized algorithm of a set of three reactive controllers shown in Figure 2.14 adopted from Pereira, Campos, and Kumar [16], the switching between these controllers is decided depending on a constraint depending on the relative position of the robot with respect to other robots and the object orientation with respect to the robots.

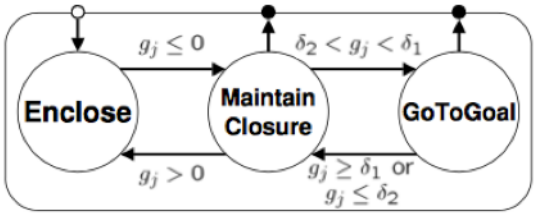

FIGURE 2.14: The three reactive controllers adopted from Pereira, Campos, and Kumar [16].

Groß, Tuci, Dorigo, *et al.* [17] used self-assembled robots for the transportation process. The controller in this paper is divided into two parts; the first part which controls the assembling of the robots, and it consists of a simple neural networks, and the second part which controls the transportation of the object to destination position and this is divided into two sub parts, the first sub part is a hand-coded algorithm to control the robots to recognize the destination position, the second sub part is a recurrent neural network that forces the motion of the robots when they can't recognize the destination position. The steps from the moment the object is found to the start of transportation is shown in Figure 2.15 adopted from Groß, Tuci, Dorigo, *et al.* [17], the robots keep trying to position around the object until they find the best

formation and they assemble in this formation and start the transportation process.

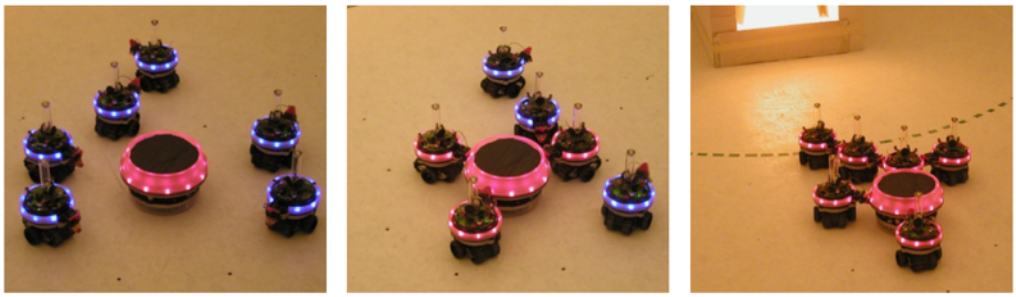

FIGURE 2.15: Self-assemble and transportation.

Fink, Hsieh, and Kumar [18] also used the caging technique in transporting the object. The controller proposed is a mix between decentralized shape controller and a global navigation function. The behavioral architecture of this system is shown in Figure 2.16 adopted in Fink, Hsieh, and Kumar [18], it consists of three states, the switching between these states is decided depending on a simple readings read from the sensors.

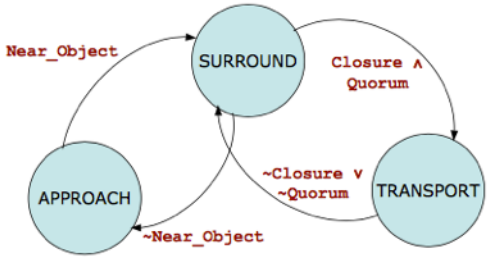

FIGURE 2.16: Behavior architecture.

Gross and Dorigo [19] used artificial neural network as the control system to control the movement of the robots where the input to the neural network is the readings from the sensors of the robot and the output is the commands needed by the motors of the robot to start movement direction. They also chose to use evolutionary algorithm in the transportation process, depending on a genetic algorithm and a fitness function to decide the weights of the neural network output.

Rubenstein, Cabrera, Werfel, *et al.* [20] suggest the use of a decentralized simple strategy to transport the object. In this strategy, each robot applies a force in the direction of the destination position without taking into consideration other forces exerted by the neighboring robots.

Sugawara, Correll, and Reishus [21] are suggesting a new technique for object transportation using the phenomena of granular convection. In this research, the researchers' system is derived from the robots after applying an external force on them,

this external force can be gravitational force or magnetic force or maybe even something larger like infrared or chemical gradients. They chose a probabilistic model for robot movements.

Wilson, Pavlic, Kumar, *et al.* [22] proposed the use of decentralized controller according to a state machine that consists of three states shown in Figure 2.17 adopted from Wilson, Pavlic, Kumar, *et al.* [22]. The transition between these states is done in a probabilistic manner depending on each state.

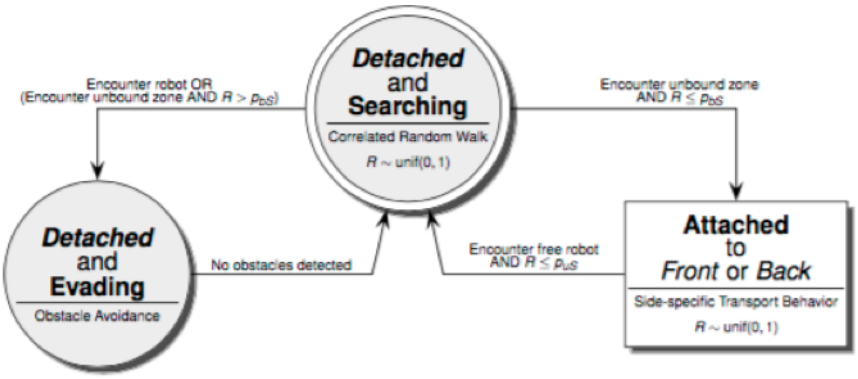

FIGURE 2.17: Three state controller.

Yu, Yasuda, Ohkura, *et al.* [23] suggested a new controller for the searching stage based on the CMA-NeuroES for the robot using artificial neural network proposed by adding complexity to the environment with the aid of covariance matrix adaptation evolution strategy (CMA-Neuro) agreed by the incremental artificial evolution method. In this paper, the researchers chose the incremental method as it helps dividing the big problem to smaller tasks making it easier for the evolution method to find sequential solution for each task.

TORABI [2] implemented an algorithm based on the procedure of food retrieval in ant coloney. In this paper, the researcher suggested the use of a simpe coordination technique between robots based on odometry with the aid of an omni-directional camera and he made sure the system is completely decentralized. In this system, the communication between the robots is in minimal state or maybe no communication at all due to the relying on the camera.

Habibi, Kingston, Xie, *et al.* [24] proposed an algorithm using the same division mentioned before, that divides the controller of the robots in to four controllers, rotating the object around a pivot robot, rotating the object in place around its centroid, translation and combing translation with rotation motion. In this paper, the researchers divide the robots into gripper robots; these are the robots that grip the object to destination, and guide robots that defines the path to destination as shown in Figure

2.18 adopted from Habibi, Kingston, Xie, *et al.* [24].

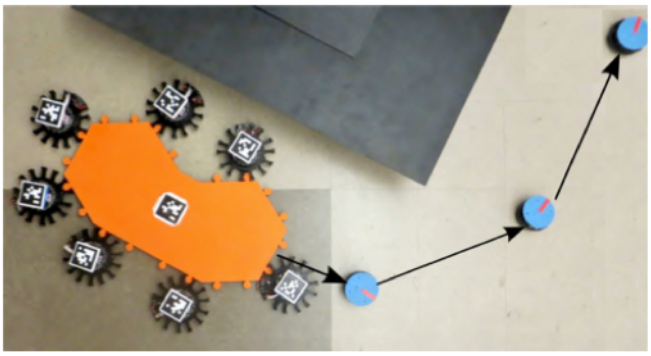

FIGURE 2.18: Guide robots (blue) and gripper robots (black).

Chen, Gauci, Li, *et al.* [25] also use a decentralized controller to control the motion of the robots, the researchers chose the state machine shown in Figure 2.19 adopted from Chen, Gauci, Li, *et al.* [25] as their controller. In this paper, they tackle the problem differently, of course the size of the object to be transported can cause occlusion to the robots and to help the robots to see the target position, but instead of seeing this as a problem they decided to use it to help the robots move to target position. The robots keep pushing the object in the direction that the object occludes by its surface, this always guarantees the object reaches destination but not using the optimal path.

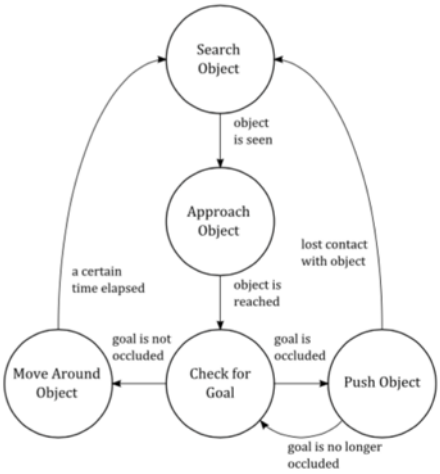

FIGURE 2.19: Controller state machine.

Habibi, Xie, Jellins, *et al.* [26] suggested to divide the object transportation task but in a different way that is shown in Figure 2.20 adopted from Habibi, Xie, Jellins, *et al.* [26], the researchers proposed dividing the robots into two, the first group is assigned the path planning task where the group explores the environment searching for the target position and constructing the best path to reach it. The second group

of robots is assigned the object manipulation task where the group is responsible for holding or gripping the object and transporting it through the environment taking into considerations any obstacles and communicating with the first group of robots to get the path already constructed to move through it.

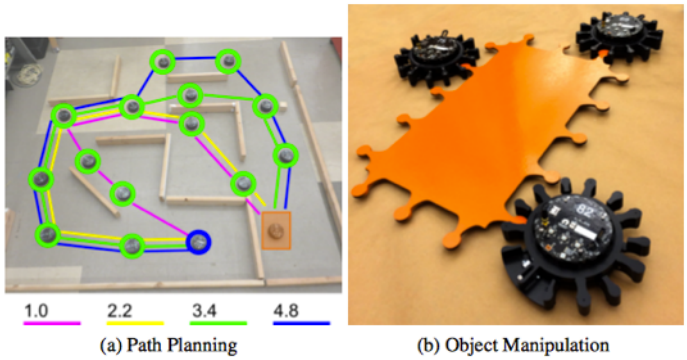

FIGURE 2.20: a. path planning robots, b. object manipulation robots.

Feo Flushing, Gambardella, and Di Caro [27] defined the STASP-HMR problem and proposed a decentralized approach for it. In this approach, each robot executes a common mathematical model depending on local information and limited sharing option. In the paper, the researchers divided the solution into two parts. The first part is proposing a decentralized coordination and planning algorithm taking into consideration both the computational and communication requirements. As for the second part, they suggested a top-down algorithm to help developing a decentralized system from the centralized mathematical model.

2.3 Conclusion

Cooperative transport in swarm robotics literature nearly collect all proposed techniques and methods in literature and briefly explain them, giving us an overall insight in the world of cooperative transportation in the literature. At the time of proposing this work, we were not able to locate any paper talking about multiple objects transportation at the same time. All available techniques in literature investigate the transportation of only one object at a time resulting in multiple techniques and algorithms in this area, but with deficiencies in the area of multiple objects transportation, leading to the introduction of the problem to be solved in this thesis, which was also discussed.

Chapter 3

Proposed Solution

In this chapter, we present our proposed solution for the problem discussed in chapter 1. The problem we are concerned with is transporting multiple objects at the same time with the aid of swarm robotics systems. As mentioned in chapter 2, this problem was not tackled nor discussed in any research yet, hence no baseline to start with, so, to start tackling the problem in hand we are setting some basic assumptions that will be discussed shortly.

3.1 Proposed System and Solution

In this thesis, we intend to construct an environment that resembles a damaged building/floor where multiple victims are trapped inside. Our environment contains multiple obstacles that are distributed randomly, as shown in Figure 3.1, this figure represents our environment with obstacles only, it was conducted on VREP simulator, the robots to be used will be ePuck robots. This application is one of the well known applications in this domain, known as search and rescue application and was discussed before and solved in multiple different ways but for only transporting one victim/object at a time.

Our robots traverse the proposed environment with the target of finding and transporting multiple victims at the same time for rescue purposes. The goal is to rescue all victims inside the building in an efficient way in the least possible time before building damage prevents safe rescue operations. The search and rescue operation must avoid any obstacles in the path. In our experiment, we start with tens of robots, they start the exploration phase, and the number of robots keeps decreasing due to the formation of smaller group of robots that will start transporting an object. In our system, the robots are divided into two types, the recruiter robot that is the robot that spots the object first and starts the recruiting process and the recruited robots that are the robots that got recruited to transport the object.

As for the algorithm for transporting multiple objects, according to Couceiro, Vargas, Rocha, *et al.* [28], the authors investigated five algorithm for searching any system,

FIGURE 3.1: Environment with obstacles only

these algorithms are Robotic Darwinian Particle Swarm Optimization (RDPSO), Extended Particle Swarm Optimization (EPSO), Physically Embedded Particle Swarm Optimization (PEPSO), and Glowworm Swarm Optimization (GSO). We work on modifying the Robotic Darwanian Particle Swarm Optimization (RDPSO) to transport multiple objects at a time, our new algorithm is called Multi Robotic Darwinian Particle Swarm Optimization (MRSDPSO). A pseudo code of RDPSO algorithm is shown in Algorithm 1.

As for the proposed system, we are using the methodology of breaking down the problem into four stages as explained by McCreery and Breed [14]. In their paper, they divided the system in to four stages, the decision stage, the attraction/recruitment stage, the organization stage and lastly, the transportation stage; the four stages were discussed earlier in literature review and summarized in Figure 2.13. The four stages are illustrated with our proposed solution in Figure 3.2. Next, we explain our proposed solution for each stage.

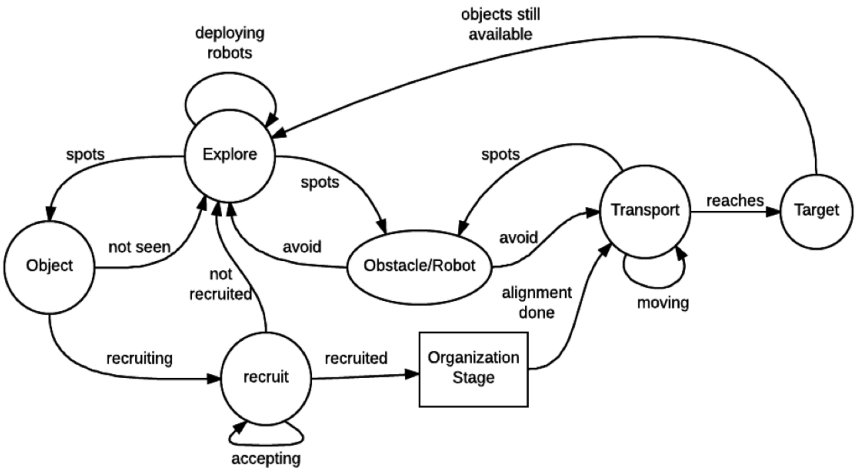

FIGURE 3.2: Proposed solution state machine

3.1.1 Decision Stage Solution

The decision stage represents the start of any swarm system. In this stage, our system consists of a group of robots that get deployed in our environment randomly, they are positioned at the start point which is going to be later the target point that the robots aim to reach to rescue the victims. The robots start moving around trying to explore the whole environment searching for any object to be transported, in the way avoiding any obstacles that are scattered around. While robots move they keep track of their movement inside the environment to be able to return with the rescued victim to the target position. In our application, the starting position for instance the door is also the target position to return to with the victims.

At this point, we have one of two states as seen in Figure 3.3, the first state which is the explore state, where the robots are still moving around exploring the environment with the target of finding an object/victim to be transported, hence, move to the next stage. The second state which is the object state, where one robot spots an object and avoid other robots from re-finding it to go to second stage which is the recruitment stage. The spotted object is no longer seen by other robots for the robots to keep searching for other objects in the environment. Later on the object is re-seen by the robots that get recruited to transport it.

3.1.2 Attraction/Recruitment Stage Solution

This stage is the second stage in our system, it is reached when a robot spots an object that needs to be transported to destination area. Once the object is spotted, at this point the target is only detected by the robot that spotted it and can not be detected

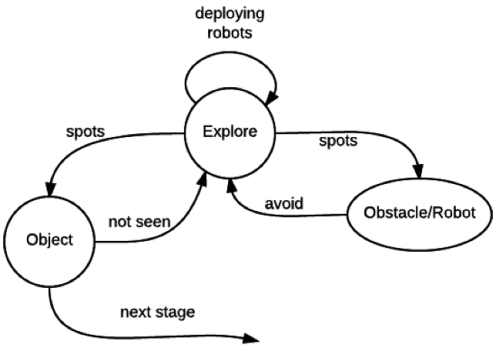

FIGURE 3.3: Decision stage state machine

by other robots in the environment to avoid any conflict of more than one robot trying to lead the transportation process. The next state at this point is to initiate the recruitment method to seek help of other robots to transport the object. The communication technique between our e-puck robots is based on zigbee, since this is the module provided with e-puck robots. There are multiple techniques for recruitment that were discussed in Chapter 2. In our system, the robot initiates communication with nearby robots to start the recruitment phase.

When the robot establishes communication with robots around it, there will be two states as shown in Figure 3.4, the first state, the recruit state where the number of robots needed for transporting the object is not met so the robot keeps accepting other robots to join its group, hence, the recruited robots start seeing the object. The second state which is the organization state, where the number of robots needed for recruitment is met, at this point if a robot is contacted for recruitment, the leading robot won't accept it joining the group so it will return to exploration mode and the whole system moves to the next stage.

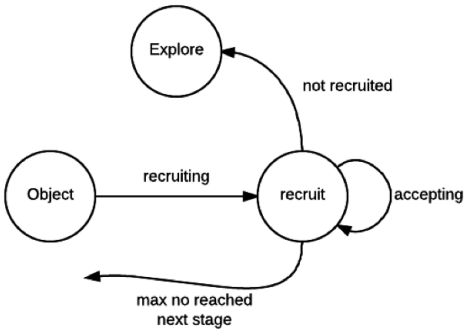

FIGURE 3.4: Recruitment stage state machine

3.1.3 Organization Stage Solution

The organization stage is the most important stage in our four stages solution. In this stage, the group of robots that is recruited to transport the object along with the recruiter robot start aligning to reach the best direction to target to be able to accurately transport the object. The aligning process mainly depends on the method of transportation. Three transportation techniques were discussed in chapter 2. We chose to use the pushing technique in our algorithm. We align the robots in the positions where the target destination is in front of them, so when they start pushing, the target moves toward the final position. We calculate the side in the target that is the farthest from the end position, and according to the number of recruited robots decide equal number of positions along this side for the robots to go to and start pushing.

3.1.4 Transportation Stage Solution

This is the stage where the main action of transportation takes place. By the end of this final stage in our solution, the object reaches the destination position. We discussed in chapter 2 three techniques for object transportation, which are pulling, pushing and encircling. The first technique is pulling the object from front and in this case the robots must get aligned in front of the object and must be equipped by grippers to be able to pull the object, the second technique is pushing the object to destination and for this, the robots need to be aligned towards the back side of the object, finally, the third technique is encircling the object to transport it, and in this case, the robots form a circle around the object and move it together. We chose the pushing technique to be our transportation technique in this thesis.

The whole group at this point works together to relocate the object toward the destination position. At this stage, we assume that the transportation path is a clear path in front of the robots, without any obstacles to avoid. The group of the robots keeps moving towards destination and once reached, they have one of two options as seen in Figure 3.5, either to go back to exploration state and keep looking for other objects to be transported, or reach a halt state as this marks the end of our experiment. If all the targets in the environment are transported to final position, then our experiment is marked as a successful one.

3.2 Swarm Robotic Environment

In this thesis, we aim to transport more than one object at a time using swarm robotics system. Our environment is divided in to two main platforms, the robots and the simulator. As for the robots used, we will use ePuck robots. ePuck robots

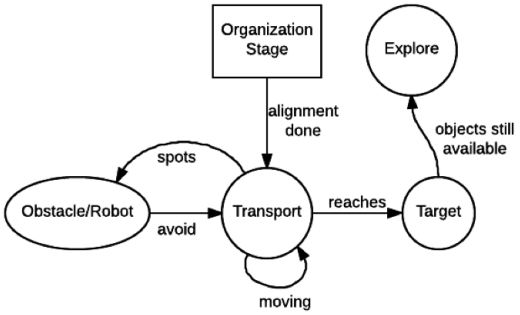

FIGURE 3.5: Transport stage state machine

are very powerful robots. We mainly chose ePuck due to its ability to move objects due to its motor. In Appendix A, there is a complete comparison between most commonly used robots in literature of swarm.

The transportation of objects is conducted in a simulated environment, for this matter we chose VREP simulator. The choice of VREP as our environment simulator is due to multiple reasons, the main reason is being an open source software and support is highly provided, along with having a complete library for ePuck robot that we use as our robot in this thesis, another comparison between available simulators in literature of swarm is located in Appendix A

3.3 Performance Measures

In this section, we discuss the performance measures that we compare our results against. Since we are unaware of any available systems similar to ours till this moment, the suggestion is to compare our system to three selected papers that are Ghosh, Konar, and Janarthanan [1], TORABI [2] , and Kube and Bonabeau [3]. In each experiment, we focus on collecting the following set of criteria:

- Environment area.
- Swarm size.
- Experiment runtime
- Success/Failure

Success/Failure of each experiment is to transport all existing objects in the environment to the target position which we define in our experiment as the door of the building. In our experiments, we fix the value for both the environment area and the number of objects to be transported and collect the data for the rest of the above mentioned criteria. Afterwards, we change the value for both the environment area

and the number of objects to be transported and recollect the data for the same criteria as done before.

We compare our proposed multiple object transportation system running MRDPSO algorithm in terms of accuracy and time efficiency to the results from three selected papers that are Ghosh, Konar, and Janarthanan [1], TORABI [2], and Kube and Bonabeau [3].

3.3.1 Accuracy

Accuracy of transporting multiple objects at the same time is actually the main purpose of our thesis and is of high importance in determining our system as a success system in swarm transportation. Accuracy is defined as the ability of the robots to transport multiple objects at a time to a specific destination. The success rate for accuracy is decided on whether the robots are able to transport the objects to the desired place.

3.3.2 Time Efficiency

Time efficiency of transporting multiple objects at a time is another important metric to be taken into consideration. Time Efficiency is the time taken by the robots to transport multiple objects at a time in comparison with the time needed to transport same number of objects by single object transport systems.

We compare our multiple objects transportation system to the results from three selected papers that are Ghosh, Konar, and Janarthanan [1], TORABI [2], and Kube and Bonabeau [3]; where the environment area will be fixed for a set of experiments, while the rest of the criteria will vary across various trials. The results of the conducted trials for each experiment will be presented in a table along with one illustrative graph that shows the relation between the swarm size and the runtime.

Chapter 4

Experimental Methodology

In this chapter, we discuss all the techniques we used, we explain them briefly and elaborate on how we implemented them. Also, we explain the algorithm we implemented, the MRDPSO algorithm in details and explain each of its stages and all the methods used within. Moreover, in this chapter, we review our experimental strategy and show our different environments. Furthermore, we compare our implemented algorithm to the results obtained from three selected papers that are Ghosh, Konar, and Janarthanan [1], TORABI [2], and Kube and Bonabeau [3]. And finally, we state the performance measures that we will compare against in our results. Our comparison is different due to the different nature of our environment than that of the papers we are comparing to. Our environment is a multi object transportation environment, while their is a single object transportation environment. In order to overcome this, we got our average time and divided it by the number of targets transported.

4.1 Methodologies

In this section, we present the techniques we used through out our experiments. We explain how each one is implemented and how each one is put in use through the experiments. Along with these techniques used, we explain our proposed algorithm, MRDPSO algorithm.

These techniques are:

1. RDPSO Algorithm.
2. Target estimation technique.
3. Recruit Technique.
4. Transportation Technique.
5. Probabilistic Finite state machine.
6. MRDPSO Algorithm.

4.1.1 RDPSO algorithm

The main idea for RDPSO is dividing the whole swarm into a set of smaller swarms. The goal of each swarm set is finding a proper solution and at the end all these solutions are to be compared and the best solution is chosen among the found solutions. Then the different swarm sets change their solution to the chosen best solution in the environment. The robot distribution depends mainly on their deployment in the environment which is chosen to be random for our experiments.

We modified the famous RDPSO algorithm which is an exploration algorithm and added to it the transportation part. RDPSO depends on having multiple groups of robots exploring the environment with the aim of finding a specific thing or mapping the environment or drawing the path between two points, we extended the functionality of this algorithm that now it can retrieve an object to a specific place. Not only one object to be transported but in our scenario, we transport multiple objects at the same time. The RDPSO algorithm pseudo code can be seen in Algorithm 1.

In our experiments, RDPSO is used, where the robots are deployed in the environment in groups. These robots then start to randomly move exploring the environment. Once a robot finds a target, it starts to estimate the size in order to estimate the number of robots that it needs to recruit to move the target. Once that is done, the robot sends signals with the position that the recruited robots need to move to in order to get ready for transportation. Once all needed robots arrive to their positions, they start pushing the target towards the end position that the target needs to reach. The main idea here is that two or more robots can find different targets at the same time. Each robot will broadcast its signal to the other robots, which aren't recruited, in order to recruit them. If the robot couldn't recruit all the needed robots needed to transport the target, then the experiment will fail in spite the fact that the robot found the target.

4.1.2 Target estimation technique

Target estimation is very important in cooperative transportation. This technique is the one that decides the number of robots needed to be recruited. In another words, the decision of the number of robots that fits to move an object depends on this technique. In our experiments, we make the robot rotates around the object in order to get the corners of the object and from these corners, the robot computes both the length and the width of the object and hence calculates its area. According to the area of the object, we decide the number of robots needed to be recruited. We made an assumption at this point that every 3 centimeters square of area need one robot to transport. So, if we have an object with width 3 centimeters and length 3 centimeters,

Algorithm 1 RDPSO Algorithm

```
1:  procedure EXPLORE_ENVIRONMENT
2:      num_of_swarms ← deploy_robots()                ▷ Robots deployment
3:      for i = 1 to num_of_swarms do                  ▷ Loop over 'i' swarms
4:          for j = 1 to num_of_robots do              ▷ Loop over 'j' robots
5:              S ← current_solution()
6:              if S > S_best then
7:                  S_best ← S
8:              end if
9:              build array X for all S_best for swarm i
10:             X_max ← max(X)
11:         end for
12:         build array B for all X_max
13:     end for
14:     for i = 1 to num_of_swarms do
15:         if B_i >= threshold then
16:             reward_swarm()                         ▷ create robot or swarm
17:         else
18:             punish_swarm()                         ▷ exclude robot or swarm
19:         end if
20:     end for
21: end procedure
```

its area will be 9 centimeters square. This object will need three robots to transport it.

4.1.3 Recruitment technique

Recruitment technique is concerned with the communication between robots. Recruitment is the start of the transportation process. In recruitment, the robot has found a target to be transported but it can't transport it on its own, so it needs the help of other robots to finalize the transportation. There are multiple techniques for recruitment in literature and we stated them in Chapter 2. The recruitment method we chose for our experiment is recruiting robots that are found around the area where the target was found. The robot that found the target starts by sending signal that can be received by any robot in explore state and waits for confirmation and number of recruited robots to reach the number of needed robots to transport the target. This is illustrated in the recruitment sequence diagram shown in Figure 4.1.

4.1.4 Transportation technique

Transportation technique is considered the main purpose of our thesis. The transportation of a target from its current position to a desired position or as in our case the transportation of multiple objects to a specific destination. In our experiments, we use pushing technique as the method for transporting the objects. The robots that

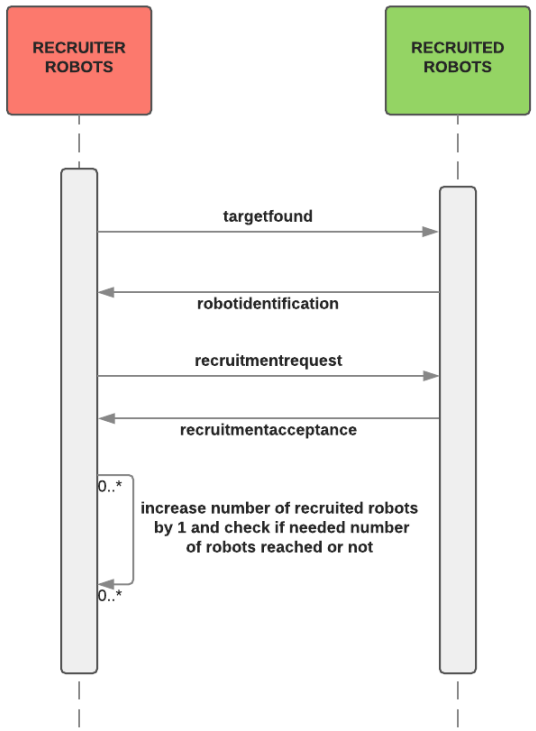

FIGURE 4.1: Recruitment Sequence Diagram

get recruited are directed to go to the side that is the farthest from the end position with orientation looking to the final position, then they start pushing towards the final position. Each recruited robot that reaches the final destination reports back to its recruiter robot that it is done with the transportation process, and the recruiter robot waits to get confirmation from all the robots that it recruited in a previous step in order to communicate in the environment that this target is transported.

4.1.5 Probabilistic finite state machine (PFSM)

Finite state machine is a mathematical representation for all the possible states a robot can be in. Finite state machine facilitates the movement of the robots in our experiments. The transition between the states takes place depending on multiple possibilities, which can be initiated from the robot itself or the surrounding robots or the different elements in the environment; i.e. if it encounters an obstacle or reaches the target.

PFSM helps in the transition of the robots from one state to another depending on a specific probability that is calculated in relation with the fired actions to move to the next state. Since in our case, the transition between the states is not binary, meaning a robots can receive more than one signal at a time. The robot uses these signals to decide depending on calculated probabilities which state it should move to.

In our experiment, any robot can take one of two paths, either the recruiting robot path which is the robot that finds the target and initiate the transportation process or the recruited robot path which is the robot that receives the recruit signal and participates in the transportation process. The finite state machine for our implemented algorithm is shown in Figure 4.2 and will be explained in details in the next section.

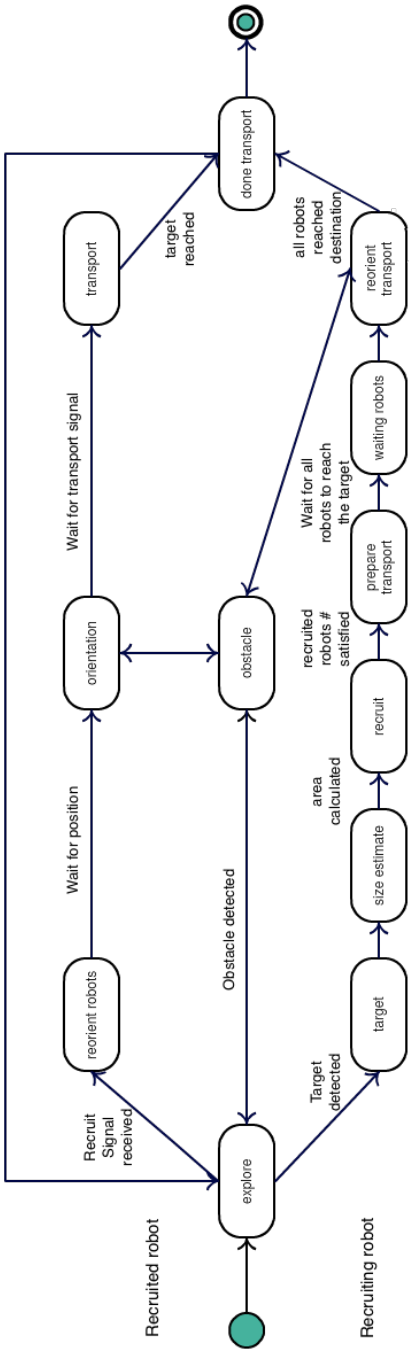

FIGURE 4.2: MRDPSO algorithm Finite State Machine

4.1.6 MRDPSO algorithm

In our proposed algorithm, we used the finite state machine methodology to define the states that the robot can go to in the environment. We have twelve states which are explore, obstacle, target, size_estimate, recruit, waiting_robots, prepare_transport, reorient_robots, orientation, transport, reorient_transport and done_transport, and are shown in Figure 4.2. The movement between states takes place according to each robot's position and the surrounding environment.

Let's start with the explore state, from the name it is the state in which the robot goes around the environment trying to locate an object that needs to be transported. This is the start state of all the robots and once they move to another state, they can return to this case in one of two cases, first case, if the robot spots a target but it was with an angle that the robot is unable to estimate its size, in this case the target is marked not found and the robot return to explore state, the second case is when the robot is done with transporting the object then it moves to explore state to start looking for remaining objects.

One more functionality happens in the explore state is listening to the recruit signal, if a robot already found a target and is done with the size estimation; the size estimation state will be discussed later; it will start with the recruiting of other robots to help in the transportation task; recruiting state is also going to be explained shortly; so the robot in the explore state is always listening for the recruit signal to move to help the other robot.

Second the obstacle state, in this state the robots avoid obstacles that they encounter while moving in the environment. The technique for object avoidance is changing the robot's orientation and send it to explore state to continue exploring the environment. This state can be reached from the explore state, the orientation state or the reorient state.

Third the size estimate state, this state is one of the important states, as in this state the robot estimates the size of the object to figure out how many robots needed to help in the transportation of that object. The technique we will use for size estimation is once the robot reaches the target, it starts rotating around the object to get its corners and from the computed corners get the length and the width of the object and hence calculates the area of the object. After size estimation the robot goes to the recruit state.

Fourth the recruit state, at this point the robot is ready to seek the help of other robots to transport the object. This is one of the states that depends on the communication between robots. The communication between the robots takes place through the wireless modules inside the robot. The robots in explore state are listening to any

recruit signal that might be sent from any robot in contact with a target object. After this, the recruiting robots move to the prepare transport stage and the recruited robot moves to the reorient robot state.

Fifth the prepare transport state, in this state the recruiting robot computes the positions that the recruited robots need to reach in order to be able to transport the object. After computing these positions, the recruiting robot communicates the positions to the recruited robots in order for each recruited robot to start moving to its specified place. Then, the recruiting robot moves to the waiting robots state.

Sixth the waiting robots state, in which the recruiting robot computes the position it needs to go to, in order to be ready to start the object transportation. It depends on the previously calculated corners in size estimate state to determine the nearest two corners to the end position. Then it uses these two corners to position itself near the middle of the object side of these corners. Once it arrives to that position, it waits for the rest of the recruited robots to reach the target in order to go to the reorient transport state.

All the previous states handle our algorithm from the point of view of the robot that found the target to be transported and wants to start the recruiting process. Now, we will list the states from the point of view of the robot that was exploring the environment and received a recruitment signal to help the recruiting robot.

First the reorient robot state, the robot reached this state as it received the signal to be recruited while being in the explore state. In this state, the robot is waiting for the recruiting robot to send the new position that the recruited robot needs to go to. Once the position is communicated to the recruited robot it sends back a confirmation signal to the recruiting robot and moves to the orientation state.

Second the orientation state, this state handles the movement of the recruited robot towards the position communicated to it from the recruited robot. This state has multiple cases that depends on the position of the robot in the environment, it keeps alternating between states until it reaches the desired position. Next, the recruited robot moves to the transport state.

Third the transport state, this is the main target of our thesis, the state in which the target gets transported from the place it is at to the destined position. In this state, the robots are aligned at the far side of the target from the final position, so the robots keep pushing in the direction of the final position until either the target reaches destination or an obstacle was found by the recruiting robot and sends a signal to reorient the recruited robot to a new pushing position. When the target reaches end position,

the robots move to the done transport state.

Finally the done transport state, this state is reached by both the recruiting and the recruited robots. It is the state that decides whether the robots need to continue exploring the environment or whether all the targets are transported and the experiment is finished.

The states discussed above can be illustrated more in Figure 4.3, this is a block diagram that represents the states we explained along with the transition from one state to another. Also, the pseudo code for our implemented algorithm is represented in Algorithm 3 which can be found in Appendix B.

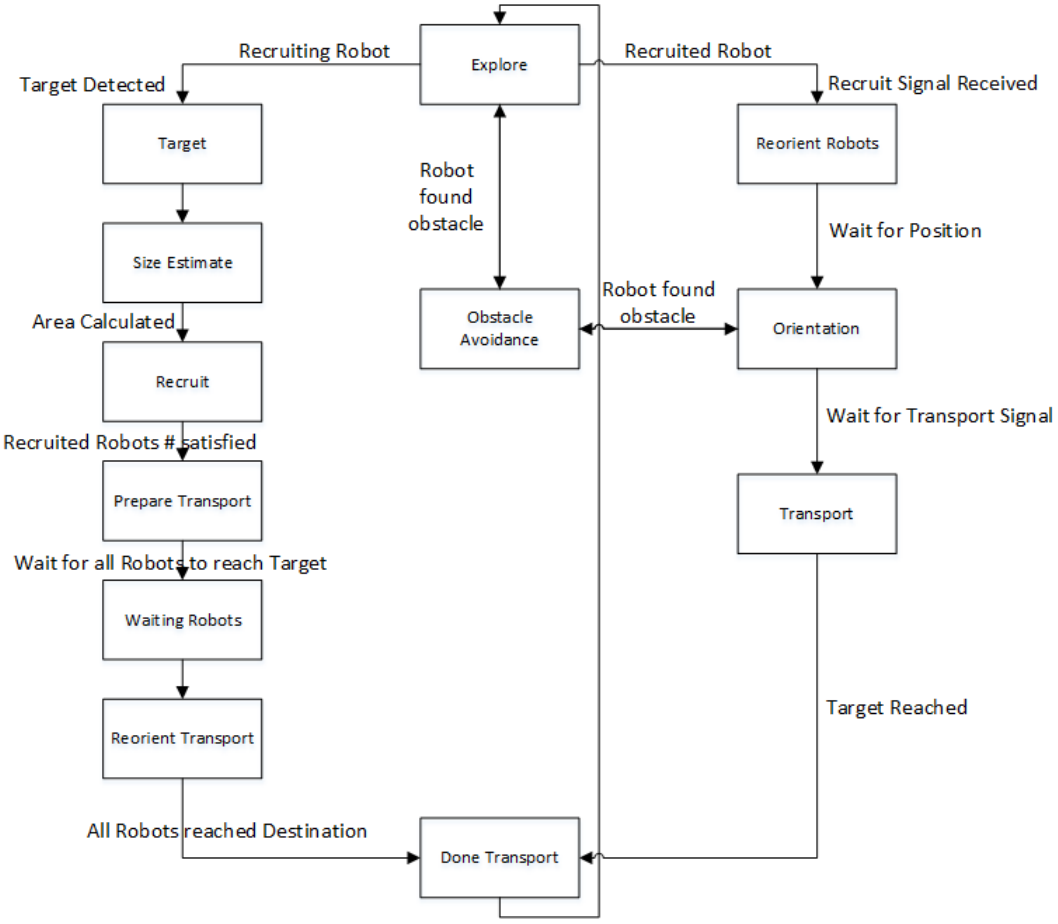

FIGURE 4.3: MRDPSO algorithm Block Diagram

4.2 Experimental Strategy

In each experiment in our thesis, we have some parameters that are constant for a set of experiments and then we vary them and start recording more results, these parameters are :

- Environment area

- Target size
- Swarm size

To assess our system, the following data will be collected in each experiment.

- Experiment runtime
- Success/Failure

Experiment runtime in our case is defined as the time needed to transport two targets from their location to the final destination. Success/Failure of each experiment is to transport the objects in the environment to the target position which we define in our experiment as the start position where the robots get deployed. In our experiments, we run a set of experiments on each area of the chosen environment areas discussed in the next section and collect the data for the rest of the above mentioned criteria.

Our test environment is divided in to two main platforms, the robots and the simulator. As for the robots used, we used ePuck robots which are very powerful robots. The transportation of objects is conducted in a simulated environment, for this matter we chose VREP simulator. The reasons for our choices for both the robot and the simulator has been discussed in Chapter 3 and the comparisons related to choosing the robot and the simulator are included in Appendix A

4.2.1 Environment Area

In our experiments, we use different rescue environment sizes as it directly affects our robot behavior. The chosen environment sizes are 50 and 100 square meters. Figures 4.4 and 4.5 represent the environments we used for our experiments.

- Medium area for our environment, 50 meters square that might resemble a floor of one damaged apartment
- Large area for our environment, 100 meters square that might resemble a floor of two damaged apartments

4.2.2 Target Size

Target size is a very important factor in our experiments, as the target size is the major contributor in deciding the number of robots needed to transport that target. Also, the shape of the target plays an important role in the size estimation of the target. In our experiments, we chose all our targets to be cuboid in shapes as we

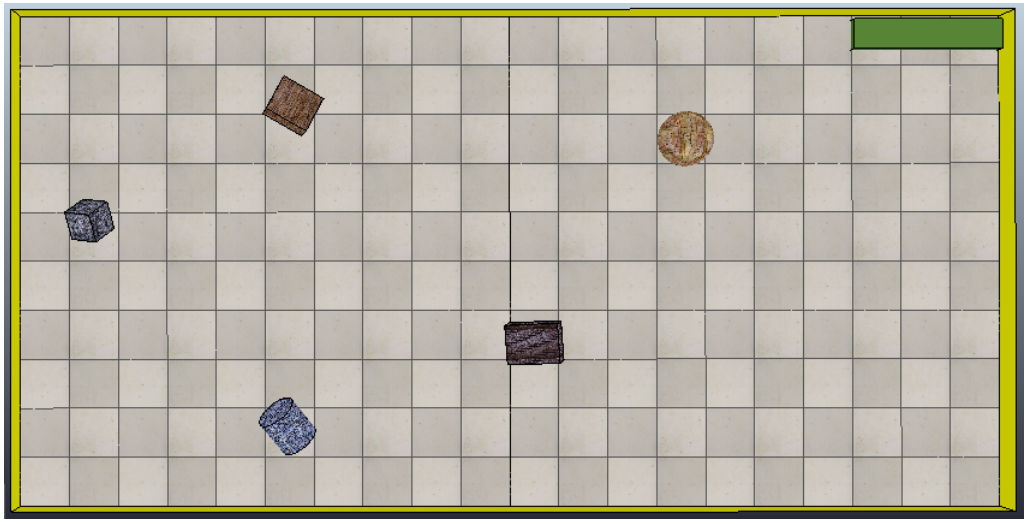

FIGURE 4.4: Environment 50 meters square with obstacles

depend on the four corners to calculate the area.

As for the weights of the target, we just added these weights depending on the size of the target. If the target is a small on, we added a small weight and as the size of the target increases, we increase the weight accordingly. For example, we have a target of size 6 centimeter squares that we added its weight to be 1.5 kilograms, a target of size 10 centimeter squares that we added its weight to be 2 kilograms and finally a target of size 12 centimeters square that we added its weight to be 3 kilograms.

In our experiments, we ran each experiment with two cuboid targets with different dimensions to ensure different area for each target and hence different number of robots will be recruited to transport each target. Figure 4.6 shows four different shapes for the targets we used while assessing our experiments. These sizes are 6 centimeters square, 10 centimeters square, 12 centimeters square, and 15 centimeters square. It was conducted using VREP simulator. We did our experiments on three target sizes only, the 6 centimeters square, 10 centimeters square, and 12 centimeters square. We manipulated the positions of the targets in each experiment in order to have more insight regarding the time needed to transport with different positions for the target.

4.2.3 Swarm Size

Swarm size is one of the variables we varied in our experiments. Swarm size affects our experiments in different ways, first way is at the beginning of the experiments, as it affects the explore state as well as the recruit state. In the explore state, swarm size affects the time taken to find targets to rescue. The second way the swarm size affects our experiments is in recruiting state as we will need specific number of

FIGURE 4.5: Environment 100 meters square with

robots to transport all the targets available in the environment, so adding number of robots less than what is needed to transport will cause our experiments to fail. We chose different swarm sizes for our experiments depending on the environment area, below are the number of robots we experimented with for the two environment areas we have:

- Environment of 50 meters square: 10, 15, 20 robots
- Environment of 100 meters square: 20, 30, 40 robots

Something worth mentioning at this point is that we randomly deploy the robots in the environment during all our experiments. Random deployment of robots in the environment ensures variety of start positions and orientation for each deployed robot and hence different results for the experiments.

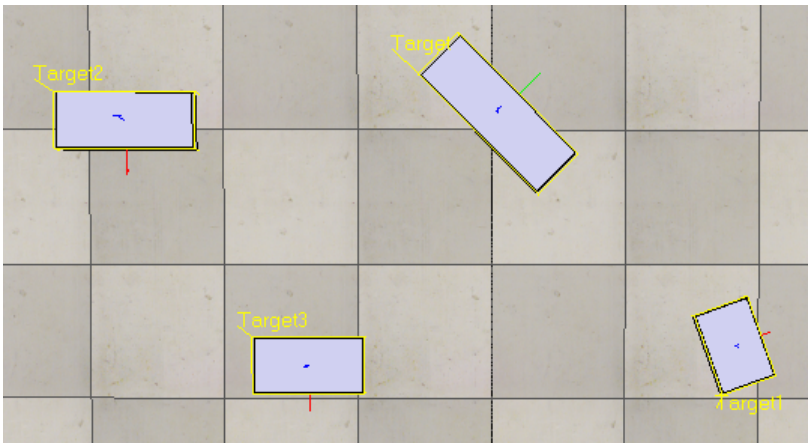

FIGURE 4.6: Different shapes for targets

4.3 Performance Measures

In this section, we define and present our metrics of comparison with the results collected from three selected papers. These papers are:

- Ghosh, Konar, and Janarthanan [1]
- TORABI [2]
- Kube and Bonabeau [3]

We define our metrics of comparison with each paper depending on the work done by the author(s), we mainly have three different metrics that are:

- Environment Runtime versus number of transportation steps
- Environment Runtime versus the mass of the target
- Environment Runtime versus the area

The comparison we conducted is a bit different since we implemented a multi object transportation algorithm and the three papers we chose are single object transportation algorithms. We solved this issue by calculating the average time needed for transportation in our experiments and divided this number by the total number of targets that got transported, and obtained the results after calculations and compared them with the results obtained from the three selected papers.

4.4 Conclusion

In this chapter, we presented the methodologies we used through our thesis. We discussed the RDPSO algorithm, the PFSM, the size estimation of the target, the

recruitment of the robots, the transportation technique we used and finally our implemented algorithm, the MRDPSO. Next, we explained our strategy in conducting the experiments, the parameters that are constant for a set of experiments and the data to be collected in each experiment along with explaining the two environments we tested on. Finally, we presented the three papers that we will compare our result to and explained the metrics upon which the comparison will be against.

Chapter 5

Experimental Results

In this chapter, we present the results of our experiments. We conducted all our experiments on simulation base, we used the VREP simulator as our simulating environment. The robot used in our experiments is the ePuck robot after altering its code with our implemented algorithm, the MRDPSO algorithm. Deciding on VREP simulator and ePuck robot was a result of a comparison made and referenced in Appendix A. We collected from each experiments the following:

- Environment area
- Target size
- Experiment runtime
- Success/Failure

All the collected data is data for transporting two targets at the same time. We discussed every metric from the above in chapter 3. In our experiments, we were able to transport more than one target at the same time, to be exact two targets and this was the main purpose of our thesis. We took more time in transporting two targets than other comparative results, but this is logical as we are transporting two targets while others transporting one target.

We ran multiple set of experiments, all on our newly implemented algorithm, varying the environment size as we had two different environments, the 50 meters square environment and the 100 meters square environment. Also, we varied the sizes of the targets between 6 centimeters square, 10 centimeters square, and 12 centimeters square. Finally, comparing our results with the results of three papers Ghosh, Konar, and Janarthanan [1], TORABI [2], and Kube and Bonabeau [3].

5.1 MRDPSO Algorithm Assessment

In this section , we will be presenting the effect of varying the mentioned metrics and will be explaining each one of them in details. The comparisons presented in this study are as follows:

- Experiment Runtime vs Environment area
- Environment Area vs Success/Failure
- Experiment Runtime vs Target size
- Single RDPSO verse Multi MRDPSO

5.1.1 Experiment Runtime vs Environment area

In this section, the relationship between the time taken to transport two targets at the same time which is referred to as the environment runtime and the environment area will be demonstrated. In chapter 4, we presented two different environment area, the 50 meters square size environment and the 100 meters square size environment. It is shown in Figure 5.1 that as the environment area increases the environment runtime increases. As it takes the robots more time to transport the target, as we increased the area they will need to cover while transporting.

In table 5.1, we present some of the data collected over a set of experiments on the 50 meters square size environment, in addition to data collected over a set of experiments on the 100 meters squared size environment that are presented in table 5.2. As seen in Figure 5.1, the minimum time taken to cover the 100 meters square size environment is 720 seconds while the minimum time taken to cover the 50 meters square size environment is 699 second.

TABLE 5.1: MRDPSO Algorithm Data collected for 50 meters square size environment

Start time target 1	Start time target 2	End time target 1	End time target 2	Status	Time difference	Time in sec
03:24	04:56	21:20	24:29	success	21:05	1265 sec
03:25	03:25	27:58	26:12	success	22:47	1367 sec
03:20	04:40	13:45	18:40	success	15:20	920 sec
02:15	06:45	31:23	45:50	success	43:35	2615 sec
0	04:22	0	22:39	fail	18:17	1097 sec
03:28	0	30:27	0	fail	26:59	1619 sec
0	04:07	0	19:40	fail	15:33	933 sec
03:07	04:41	11:43	14:64	success	11:39	699 sec

5.1.2 Experiment Area vs Success/Failure

In this section, we will present the relationship between the different environment area we have and their status after the experiment, whether the two targets reached the final position and we refer to this as a transportation success process or if one or both did not reach the final position and we refer to this as a transportation fail

TABLE 5.2: MRDPSO Algorithm Data collected for 100 meters square size environment

Start time target 1	Start time target 2	End time target 1	End time target 2	Status	Time difference	Time in sec
04:11	08:47	20:24	32:10	success	27:59	1679 sec
02:07	04:39	19:00	27:50	success	25:43	1543 sec
04:54	08:05	29:49	51:10	success	46:16	2776 sec
04:33	0	29:20	0	fail	24:47	1487 sec
03:19	0	29:20	0	fail	26:01	1561 sec
07:40	07:00	16:40	19:00	success	12:00	720 sec
03:02	06:00	27:00	23:00	success	23:57	1437 sec

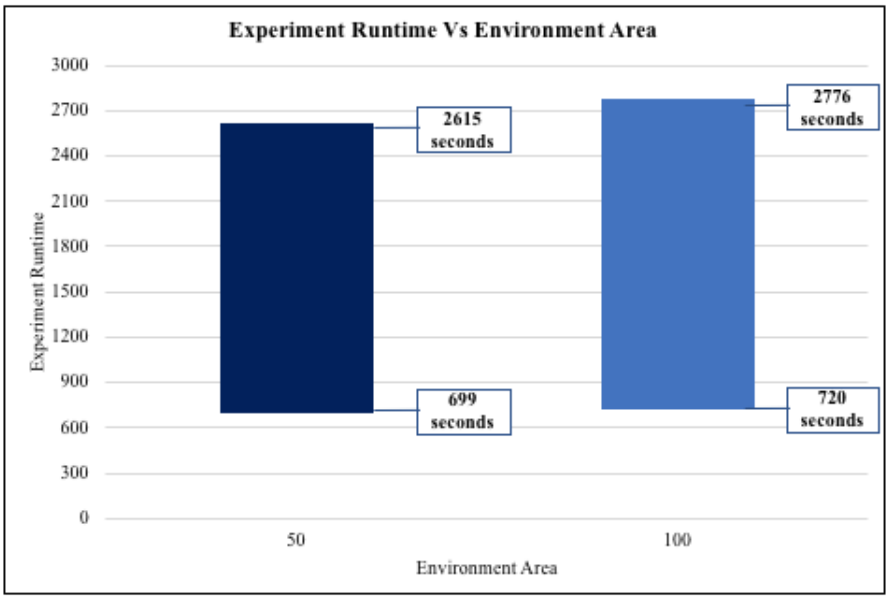

FIGURE 5.1: Experiment Runtime vs Environment area Graph

process as shown in Figure 5.2. As mentioned before, we have two different environment areas.

We had multiple failure cases, We had multiple failure cases and one of the reasons for the failed scenarios that relates to this section is the failure in the obstacle avoidance method, this failure causes one or more of the recruited robots to not reach the target they need to transport, resulting in that the reached number of recruited robots to be less than needed to transport the target and hence failure of the transportation process.

Another failure case is due to the random deployment of the robots, deploying the robots on a wide area in the environment caused the robots to take more time to reach the target and more time to transport it and in some cases they fail to transport. We concluded from that that we need to deploy our robots on not more than

20% of our environment area, so we need to deploy them in an area of 10 meters square for the 50 meters square environment area and 20 meters square for the 100 meters square environment area.

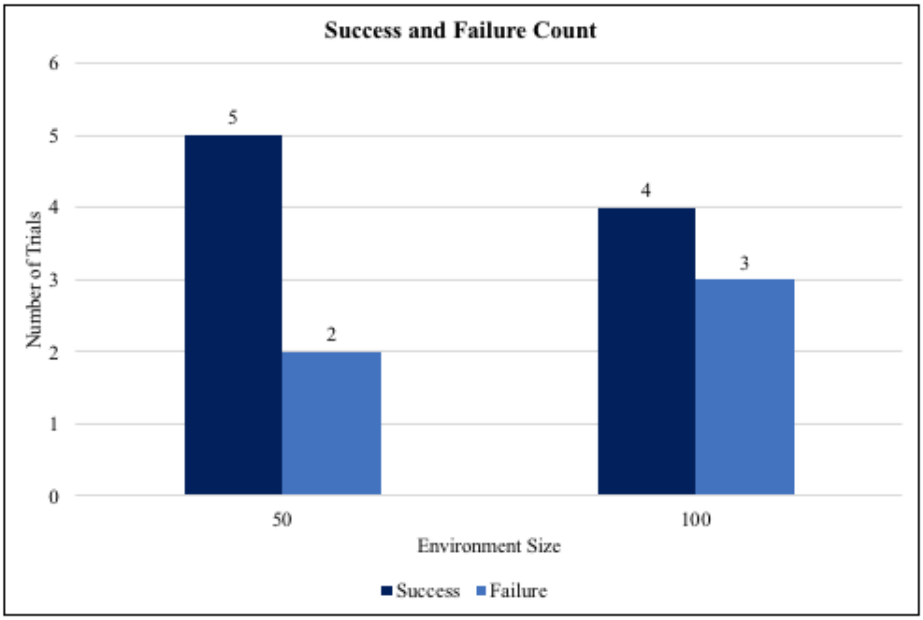

FIGURE 5.2: Environment area vs Success/Fail Graph

5.1.3 Experiment Runtime vs Target size

In this section, we will present the relationship between the time needed to finish the transportation process which is the experiment runtime and the size of the target. We varied the size of the target to test multiple used methods in our algorithm. Since the estimation of the size method, and the number of recruited robots method both depend on the size of the target. We use three different target sizes for our experiments, the 6 centimeters square size, the 10 centimeters square size and the 12 centimeters square size. In our environments, we transported a total area of 16 centimeters square, 18 centimeters square and 22 centimeters square.

During our experiments, different sizes were estimated and different number of robots got recruited depending on the size, as mentioned before, we made an assumption that each robot can transport 3 centimeters square of area, hence, in the case of the 6 centimeters square size, the recruited robot recruited two more robots to help with the transportation process. In the case of the 10 centimeters square size, the recruited robot recruited three more robots to help with the transportation process. And finally, in the 12 centimeters square size, the recruited robot recruited four more robots to help with the transportation process.

We ran this type of experiment on a set of our conducted experiments, the different sizes of the targets we chose were transported most of the time, as shown in Figure 5.3.Also, in Table 5.3 we present a sample of the collected data, these data reports the different transportation time for the summation of the area of the two targets transported at the same time which are the 18, and the 22 centimeters square. In Figure 5.3, the data for the 18 centimeters square is collected from the experiments ran on the 50 meters square environment and the data for the 22 centimeters square is collected from the experiments ran on the 100 meters square environment.

TABLE 5.3: MRDPSO Algorithm Sample collected Data for target size

Target Size	18 cm2	22 cm2
Time	21:05	27:59
Time	22:47	25:43
Time	11:39	23:57

FIGURE 5.3: Experiment Runtime vs Target Size Graph

5.1.4 Single RDPSO verse Multi RDPSO

In this section, we present the relationship between the time needed to finish the transportation process in case of a single RDPSO algorithm and the time needed to finish the trasportation process in the case of a multi RDPSO algorithm. In our experiments, we ran the single RDPSO algorithm with two targets in the environment, the robots start by fetching the first target and transporting it to final destination and then move to the second target to transport it. Then we ran the multi RDPSO algorithm with the same environment criteria, but our algorithm will fetch the two targets at the same time.

Table 5.4 represents a sample data collected for the single RDPSO algorithm and the Multi RDPSO algorithm under the 50 meters square environment, in addition, table 5.5 represents a sample data collected for the single RDPSO algorithm and the Multi RDPSO algorithm under the 100 meters square environment. Also, Figure 5.4 represents the data collected in Table 5.4, from the figure we can deduce that the multi RDPSO algorithm is 41 percent faster than the single RDPSO, and this comes from the overlap in time while transporting multiple objects in the MRDPSO algorithm while in the single RDPSO, the two targets are transported in two different time intervals. As for Figure 5.5, it represents the data collected in Table 5.5, the same is applied for the 100 meters square environment, but from the figure we deduce that the MRDPSO is 35 percent faster than the single RDPSO.

TABLE 5.4: Single RDPSO vs Multi RDPSO Sample collected Data for 50 meter square environment area

Single RDPSO	Single RDPSO in sec	Multi RDPSO	Multi RDPSO in sec
17:56 19:33	1076 sec 1173 sec	21:05	1265 sec
24:33 22:47	1473 sec 1367 sec	22:47	1473 sec
10:25 14:00	625 sec 840 sec	15:20	920 sec
29:08 39:05	1748 sec 2345 sec	43:35	2615 sec
08:36 10:05	516 sec 605 sec	11:39	699 sec

TABLE 5.5: Single RDPSO vs Multi RDPSO Sample collected Data for 100 meter square environment area

Single RDPSO	Single RDPSO in sec	Multi RDPSO	Multi RDPSO in sec
16:13 23:23	973 sec 1403 sec	27:59	1679 sec
16:53 23:11	1013 sec 1391 sec	25:43	1543 sec
24:55 43:05	1495 sec 2585 sec	46:16	2776 sec
09:00 12:00	540 sec 720 sec	12:00	720 sec
23:57 17:00	1437 sec 1020 sec	23:57	1437 sec

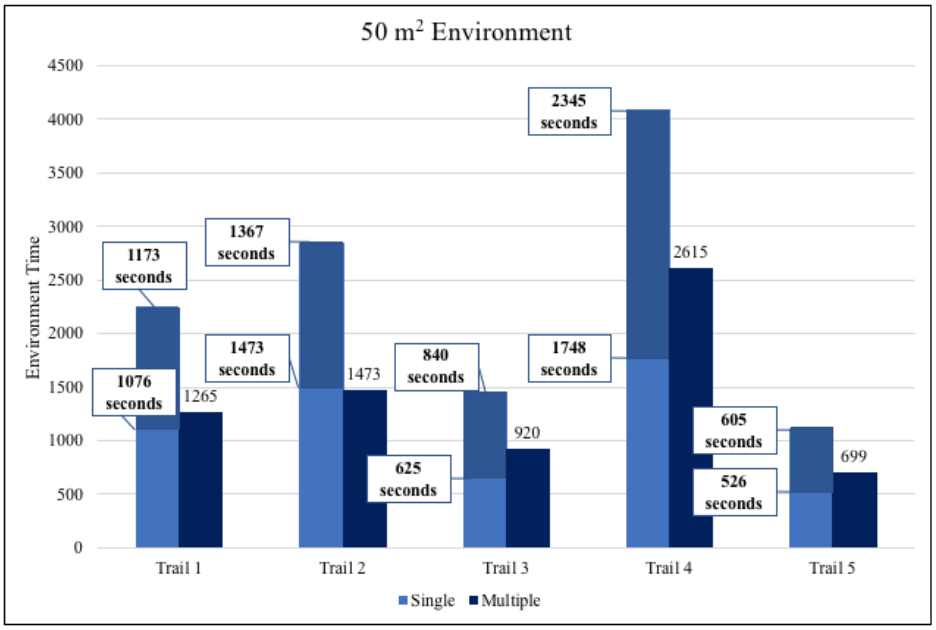

FIGURE 5.4: Single RDPSO vs Multi MRDPSO 50 meter square environment area

5.2 Performance measures

In this section, we entail our comparison with the results collected from three selected papers. These papers are:

1. Ghosh, Konar, and Janarthanan [1]
2. TORABI [2]
3. Kube and Bonabeau [3]

We already defined our metrics of comparison with each paper in Chapter 4. We chose one metric per paper, these metrics are:

- Environment Runtime versus number of transportation steps
- Environment Runtime versus the mass of the target
- Environment Runtime versus the area of the target

As we discussed in Chapter 4, the conducted comparison is a bit different since we implemented a multi object transportation algorithm while the three selected papers use single transportation algorithms. We calculated the average time needed for transportation in our experiments and divided this number by the total number of objects that got transported, and compared the new calculated results with the results obtained from the three selected papers.

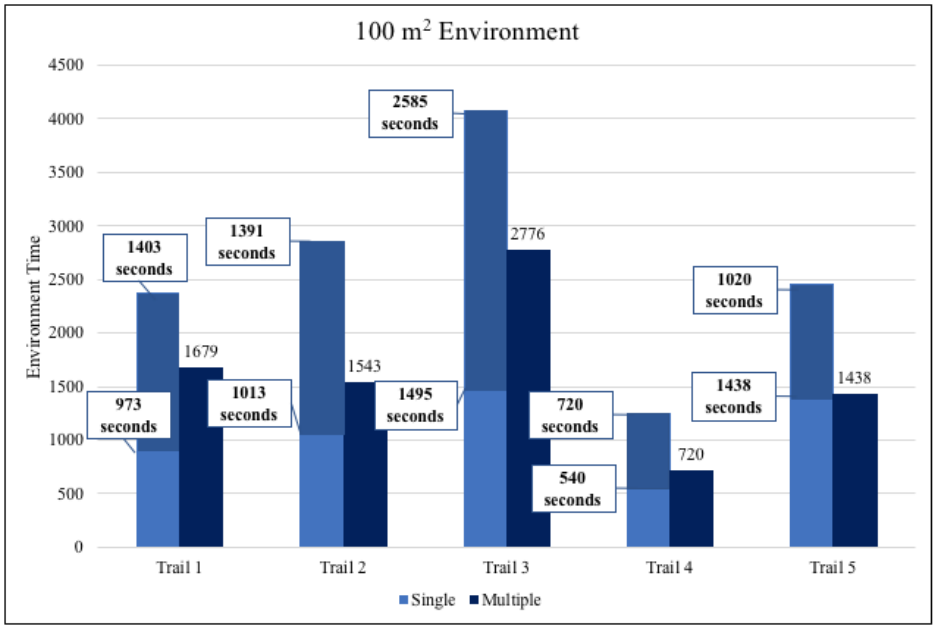

FIGURE 5.5: Single RDPSO vs Multi MRDPSO 100 meter square environment area

5.2.1 Comparison with Ghosh, Konar, and Janarthanan [1]

This paper is concerned with the problem of box pushing from its position to a target position. It depends on two similar robots having to decide on box's trajectory of motion taking into consideration avoiding a static number of obstacles in the environment. The robots in this case have the ability to change the position of a large box from its current position to the target end position. The process of changing the box position contains two main functions, turning and translation. In the first function, turning, the robots push and pull the box while in the translation; the robots only push the box. The technique used in this paper depends on both the two robots standing at the same side of the box they need to push and then both apply perpendicular forces on it. In order to avoid obstacles in the pushing direction, a penalty function is used.

We will compare our algorithm to this algorithm along with the algorithm the authors used to compare their work and results to. Figure 5.6 shows the two maps used by the authors where a box is pushed using the proposed algorithm to go all the way till the end point. In the left figure, the number of steps used to reach the target position is ten steps while in the right figure, it needed 13 steps to do the job.

We are comparing our results which are the results of a multi object transportation system with the results of a single object transportation system. In order to make it fair, we will calculate the average time needed for transportation in every environment and then divide it by the number of transported objects which is in our case 2. This roughly gives us the average time needed to transport one object with our

implemented algorithm.

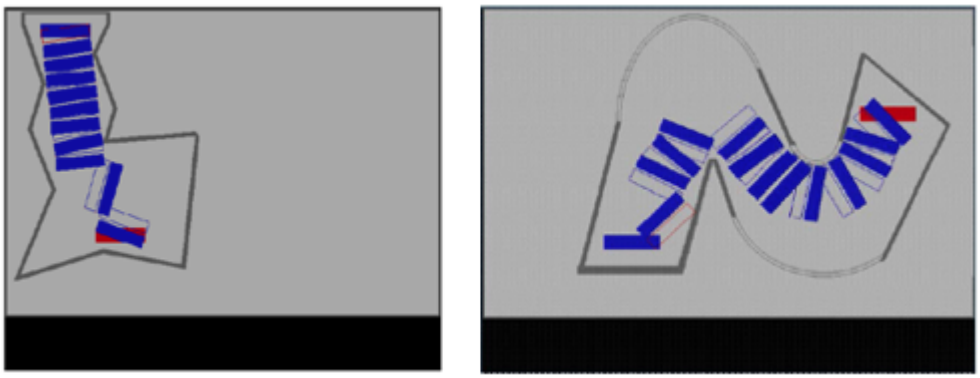

FIGURE 5.6: World maps from Ghosh, Konar, and Janarthanan [1]

In their work, they calculated the next position for the target taking into consideration the walls and then moved the target to the new position, they considered this as the step to transport the object, so the robots stopped after reaching the calculated position, calculate a new position then move the target to it. In our case, we assume the robots stop 2 times per cube, to make sure they are moving in the right direction and make sure their orientation is correct.

For the 50 meters square environment, we placed our targets, one target is placed 5 cubes away from the end position and the other target is placed 6 cubes away from the end position which maps to 10 and 12 steps respectively. As for the 100 meters square environment, one target is placed 6 cubes away from the end position and the other target is placed 7 cubes away from the end position which maps to 12 and 14 steps respectively.

Table 5.6 shows the comparison between the new calculated results for our algorithm and the results obtained from the paper, we added the numbers obtained in the paper for both the NSGA-II algorithm and the MOPSO algorithm and next added our results to the MRDPSO algorithm. All the values for the time are presented in seconds and our results are highlighted in red in Table 5.6.

5.2.2 Comparison with TORABI [2]

In order to evaluate the collective transport strategy in a 2D planar environment, a decentralized algorithm was implemented in a simulation environment by Sina Torabi, on small mobile robots' platform, where he used the same four phases we used and discussed in cooperative transport that are

- Decision phase

TABLE 5.6: Comparison between our results and the obtained results from Ghosh, Konar, and Janarthanan [1]

World Map	Method	Total Time	Steps/ Environment
1	NSGA-II	696.57 sec	11
	MOPSO	649.51 sec	10
	MRDPSO	460 sec	11
2	NSGA-II	857.94 sec	13
	MOPSO	810.36 sec	13
	MRDPSO	719 sec	13

- Recruitment phase
- Organization phase
- Transport phase

Individual trial was considered successful if robots succeed to transport the target within a certain time threshold. The threshold varied with the mass of the target in such a way that for transporting a light target, robots have less time than when they are transporting a heavy target. In addition, the study key findings show that the mass of the target slightly decreased the efficiency, indicating that team coordination is more difficult when dealing with a heavier object, which could be resolved using more robots.

More than 90% of the transportation were carried out in time. In order to compare the effect of the recruitment process on transportation time, for each target's mass, 50 trials were made, 25 trials with recruitment process and 25 without it. The experiment results indicated that in order to transport the object in more specific period of time, it's recommended that the robots use the recruitment process. Besides, results showed that a heavier target required more time to be transported and its path was longer. When the target is heavy it will only move when a sufficient number of robots are pushing it in the same direction.

In our experiments, we depend on the recruitment technique to get more robots to help in transporting the target, so we will compare with the author taking into consideration the data for the recruitment strategy, as discussed in previous section. Since we implemented an algorithm for multi object transportation and this paper also is concerned with single object transportation, we will compute our average time and divide it by two, that resembles our number of targets. In this case, we will also divide the total mass of the targets transported by the number of transported target that is 2 to get the average weight transported. Table 5.7 shows the results of our implemented algorithm in transporting a target of size nearly 2 kilograms and another target of size nearly 2.5 kilograms against the results obtained from the paper that are for two targets one of size 3 kilograms and the second 4.5 kilograms. All

the values for the time are presented in seconds and our results are highlighted in red in Table 5.7

TABLE 5.7: Comparison between our results and the obtained results from TORABI [2]

Target Mass	Strategy	Average Time
2 kg	Recruitment Method	531 sec
2.5 kg	Recruitment Method	675 sec
3 kg	Recruitment Method	186.7 sec
4.5 kg	Recruitment Method	331.8 sec

5.2.3 Comparison with Kube and Bonabeau [3]

In the experiment done by C. Ronald Kube, Eric Bonabeau for cooperative transport by ants and robots to compare the efficiency of transporting multi objects as a function of system size and object geometry in two experiments. The authors divided their whole experiment into three phases:

- Finding the box
- Move to the box
- Push to the goal

The first phase begins with the robots executing find-box and quickly disperses in the environment. Shortly thereafter, those robots that were facing the box and sufficiently close would move towards and make contact with a box side using the move-to-box controller.

In the second phase, some of the robots incorrectly positioned for pushing, as determined by the push-to-goal controller, begin moving counterclockwise around the box perimeter searching for an open spot on a correct side. The obstacle avoidance behaviors keep a robot away from occupied positions on a box side.

In the third phase, the box moves towards the goal position. Once a net force sufficient to move the box occurs, the box begins to translate and possibly rotate. During the box movement phase a robot continuously determines if it remains on the correct side for pushing. A robot located at the edge of the pushing swarm may suddenly lose sight of the goal and begin repositioning. The resulting drop in pushing force may be sufficient to halt the box movement until another robot joins the group effort.

The dynamics of both the box and the robots are such that the path taken by the box towards the goal is seldom straight. Rather, the box movement process can be said to converge towards the goal since its trajectory is the net result of several force

vectors applied by individual robots.

We will compare our results against this paper with respect to the area of the target and environment runtime taken to transport the target. In this paper, the authors used 4 different box types, Box A is of size 42 centimeters square, Box B is of size 84 centimeters square, Box C is an extension of Box A by adding a second frame and Box D is a rounded object of diameter 84 centimeters. To conduct their experiments, they used a total of six robots, each of diameter 18 centimeters.

In our experiments, we have three different sizes for the targets we used. A 6 centimeters square cuboid, a 10 centimeters square cuboid and a 12 centimeters square cuboid. Our ePuck robot is of diameter 7 centimeters. Through the transportation of the targets, we used a variety of three to six robots to transport each target. As mentioned in the two previous comparisons, in order to be able to compare our multi object transportation results to their single object transportation results, we will take the average time for our experiments and divide it by two. In order to get the average size of the target transported we will also divide the total weights of the targets by two.

We will only compare with Box A and Box B, since Box C is an extension of Box A but we do not have the its exact area and Box D is a rounded box and all our targets are cuboid targets. Table 5.8 shows the results of our implemented algorithm in transporting a target of area nearly 9 centimeters square and another target of area nearly 11 centimeters square against the results obtained from the paper. All the values for the time are presented in seconds and our results are highlighted in red in Table 5.8.

TABLE 5.8: Comparison between our results and the obtained results from Kube and Bonabeau [3]

Target Area	Average Time
9 cm2	531 sec
11 cm2	675 sec
42 cm2	around 180 sec
84 cm2	around 110 sec

5.3 Challenges

In this section, we will explain the challenges we faced in order to implement our MRDPSO algorithm and how we were able to solve them. The challenges we faced were:

1. Estimating the size of the target

2. Recruiting the needed number of robots
3. Dispatching of the robots
4. Communication between the robots
5. Positioning the robots to push towards the final destination

5.3.1 Estimating the size of the target

This challenge was one of the medium sized challenges that faced us through our implementation of the algorithm, estimating the size of the target in order to decide the number of robots needed to be able to transport it. We first thought of rotating around the body of the target and calculating the distance covered from both the speed of the robot and the time taken to rotate around the target, this method failed for two reasons, first reason is the robot was unable to find its start position so it kept rotating around the target and second reason, even if succeeded we would have been able to calculate the perimeter of the target which will not be of use to get the area of the target.

The second method we decided to try is the method we used in our thesis to calculate the area of the target and from which with the assumption that every three centimeters square of area can be transported by one robot, we were able to calculate the number of needed robots to transport the target. In this method, we left the robot to rotate around the target and collect the positions of the corners of the target, and with the assumption that we will use cuboid targets, we were able to calculate the distance between first and second corner to get the length, and calculate the distance between the second and the third corner to get the width. Knowing the length and the width, we were able to calculate the area and hence the needed number of robots for transportation.

5.3.2 Recruiting the needed number of robots

Recruiting the needed number of robots is a crucial task in transport. Without the extra number of robots, the robot that spotted the target will not be able to transport it alone and hence no transportation can take place. Recruitment can be done in one of two ways, the first way is after the robot spots the target to be transported it returns back to the start position and recruit more robots there and then moves back to the target to start transporting it or the robot spots the target and start recruiting the robots in the area surrounding the target.

In our experiments, we decided to go with the second option. It was challenging synchronizing this type of communication between the different robots. We started

by sending recruit signal from the recruiting robot to the available robots in the environment and wait for the reply signal from the robots to be recruited. At the beginning this was not enough, as we did not have any type of identification, so we were receiving multiple signals without knowing which signal belongs to which robot, hence no way to know which robot got recruited. So, we started sending global signal to recruit and each robot replies back with its identification. Second issue we got was that the robots got recruited for both targets since we did not specify any identification to the target, So, we added the target identification to the sent signal.

5.3.3 Dispatching of the robots

The dispatching of the robots was another issue that faced us. By orientation of the robots we mean, after recruitment is established, we need the robots that got recruited to go to the position of the target. Orientation can be done in several ways. In our experiments, we first decided to move the robots to the first position retrieved by the robot that spots the target. This was not successful as VREP simulator does not have any function that makes the robot move to a specific location. So, we decided to use the orientation function available in the simulator. And this was not also easy due to the difficulty presented in handling the orientation of the robot inside the simulator just provides the Euler angles.

At this point, we decided that we will implement our own function 'orient_robot' to handle the movement of the robots to the positions we desire along with readjusting the orientation of the robots. In our function, we have four cases that the robot can be in any of them and we move from one case to another, until the robot reaches the desired position. The four cases are:

- The robot is at a position where the x point and the y point are larger than the points where the desired position.
- The robot is at a position where the x point is smaller than the x point of the desired position and the y point is larger than the y point of the desired position.
- The robot is at a position where the x point is larger than the x point of the desired position and the y point is smaller than the y point of the desired position.
- The robot is at a position where the x point and the y point are smaller than the points where the desired position.

These cases are illustrated more in Figure 5.7 and Algorithm 2 shows the pseudo code for the implementation of our function as for the rest of the cases, they are straight forward, the robots just move in straight line towards the target. The straight line is decided according to the position of the robot with respect to the target.

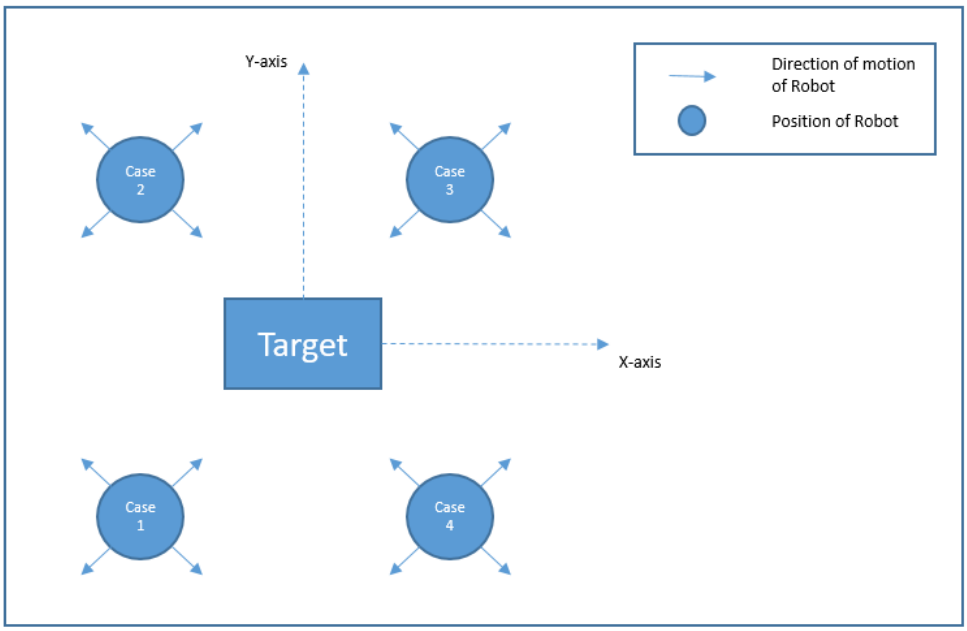

FIGURE 5.7: Orient Robot Function

Algorithm 2 Orient Robots Function

```
1:  procedure ORIENT_ROBOTS(CURRPOS, PREVPOS, LEFTVEL, RIGHTVEL, CASE)
2:      wait ← 1
3:      if case == 1 then                          ▷ Check cases in Figure 5.7
4:          if currpos.x > prevpos.x then          ▷ Compare positions to get direction
5:              if currpos.y > prevpos.y then
6:                  velLeft ← leftvel              ▷ Decide robot speed according to direction
7:                  velRight ← rightvel
8:                  wait ← 0
9:              else if currpos.y < prevpos.y then
10:                 velLeft ← - leftvel
11:                 velRight ← rightvel
12:             end if
13:         else if currpos.x < prevpos.x then
14:             if currpos.y > prevpos.y then
15:                 velLeft ← leftvel
16:                 velRight ← - rightvel
17:             else if currpos.y < prevpos.y then
18:                 velLeft ← - leftvel
19:                 velRight ← rightvel
20:             end if
21:         end if
22:     else if case == 2 then
23:         // same as case 1 but with adjusting speed according to position
```

Algorithm 2 Orient Robots Function (continued)

```
    else if case == 3 then
        // same as case 1 but with adjusting speed according to position
    else if case == 4 then
        // same as case 1 but with adjusting speed according to position
    end if
    return wait
end procedure
```

5.3.4 Communication between the robots

Communication between the robots was one of the challenges we knew we will face from day one. Handling a huge number of robots and the communication between them can be very challenging. In our experiments, since we are using simulator in our case VREP simulator, we used the concept of the signals. We overcome the problem of communication by sending signals between robots. One more important thing to communicate through any communication between robots is the target to be transported identifier, as we have multiple targets that can be transported at the same time. So, to avoid conflicts, we send the signal with the target identifier and each robot checks whether the communicated target is the one the robt is recruited to transport or not.

If we want to send some global signal, we do not need to add any identifiers to the robots to send to. While if we want to send a specific signal to a specific robot, first, we need to communicate this robot's identification to the robot controlling the recruitment process for example at the start of the communication between them, and the robot responsible for the transportation process afterwards sends the signal with the robot identification and each robot checks whether this signal is related to it or not. The kind of signal that targets specific robots are the signal to inform a specific robot that it is recruited for a specific target and the signal with the position to move to near the target to be ready for the transportation process.

5.3.5 Positioning the robots to push towards the final destination

Positioning the robots to push towards the final destination was one of the huge challenges that faced us. As mentioned earlier controlling the orientation of the robot from VREP was not easy and we had to invent our own logic to do so. In order to do this, after the robots reach the position they were supposed to reach, they start rotating in order to face the target. This is performed by using the sensors placed on the robots, checking the distance between the sensors and the target and reaching the position in which the two sensors at the front of the target reading the same distance. At this point, the robot stops its movement and waits for a signal to

start transporting the target, this signal is sent by the robot responsible for the transportation process, and it sends this signal when all the robots that got recruited are ready for transporting the target.

We have another way of positioning for the robots to push in the right direction, and that is deciding which side the robots should go to in order to start pushing the target or in another words, the best suited place to proceed with the transportation process. To accomplish that, we use the corners calculated by the robot that is responsible for the transportation process, we checked the distance between each corner and our end position and decided to make all the robots move towards the farthest side and orient themselves to look to the target so that when they start pushing, they are pushing in the direction of the end position.

5.4 Conclusion

In this chapter, we presented the results we collected from all our experiments, running simulation on the VREP simulator using the ePuck robot as our transporting robots. We were able to transport two objects at the same time which was the target aimed to be reached by this thesis. Then, We compared our results with the results obtained from three selected papers that are Ghosh, Konar, and Janarthanan [1], TORABI [2], and Kube and Bonabeau [3]. Finally, we explained the challenges we faced throughout our work and discussed how we overcame them.

In our comparison with the first paper, we were able to transport our objects in time less than the time needed by the two algorithms presented in the paper. In the second paper, our results were much higher than those obtained from the paper although we are transporting less weight objects, but this is due to the different robotic nature between our robots and the robots used in this paper, the author uses five robots each of weight 3 kilograms to transport a 3 kilograms object, while we use from three to six robots each of weight 200 grams to transport two targets of weight 1.5 kilograms and 2 kilograms. Finally, the third paper, during our comparison, our results were also higher that those obtained from the paper, but we can say this is due to the variation in the robot size since they are using six 18 centimeters robots while we are using from three to six 7 centimeter robots.

Chapter 6

Conclusion

Swarm robotics has been gaining momentum in the past decade. It originates from nature by investigating behavior of some insects like ants and bees. Ants use their technique which is leaving traces of pheromones while bees have their technique which is dancing. In this thesis, we have been working with cooperative transportation with the aid of swarm robotics. We discussed the problem at hand which is transporting multiple objects at a time. Going through literature, we were able to find multiple algorithms that investigate cooperative transportation using swarm robotics.

We implemented a novel algorithm for the matter at hand based on the RDPSO algorithm to accommodate transporting multiple objects at a time, hence calling our algorithm MRDPSO. We conducted the experiments and collected the results using the VREP simulator to simulate our environment along with using ePuck robots as the transporting robots. We compared the results of our MRDPSO algorithm with the results provided by three published papers that are Ghosh, Konar, and Janarthanan [1] which we compared with in terms of time and steps taken to transport the target, TORABI [2] which we compared with in terms of time and mass of the target, and Kube and Bonabeau [3] which we compared with in terms of time and area of the target.

We were able to achieve the aim of this thesis, which was multi object transportation. We were able to transport two targets at the same time to the final position. We presented some comparisons to some experiments on our system and our implemented algorithm. We were able to learn from the failures of the experiments in order to avoid failed experiments. We also compared to the three selected papers and were able to get better transportation time than that obtained by Ghosh, Konar, and Janarthanan [1], but got results much higher than those obtained by TORABI [2] and Kube and Bonabeau [3].

6.1 Future Work

In this section we will discuss the enhancements that we can do in the MRDPSO algorithm. These enhancements are:

- Estimating the size of the target technique enhancement in order to get better results for the target size and help with decreasing the time to get the area of the target. Size estimate can be enhanced by using different technique rather than what we used, like using centroid size estimation
- Signaling system enhancement in order to help with the communication between the robots. One thing that needs enhancement in signaling is when the recruiter robots starts to recruit other robots to take into consideration their distance from its position and recruit those who are near it rather than sending a signal that can be received by any robot in the environment
- Transportation technique enhancement in order to speed up the transporting process, one technique worth trying is for the robot to push the target for number of seconds and to stop to assess its position with respect to the target and maybe reorient itself in better position to continue pushing
- Obstacle avoidance technique while transporting enhancement in order to avoid obstacles while transporting the target to the end position
- Weight variation to try and reach the maximum weight that can be transported by our implemented algorithm.
- The number of transported objects can be added as a new parameter in running experiments and collect the same data again, then compare the new results with the results already obtained by adding the number of transported objects as a new metric for the comparison.

Appendix A

Platform Determination

Appendix A will discuss our working environment. As seen in literature there are two ways to experiment on swarm of robots, either using a simulator program that you deploy any amount of robots on, run your experiments and get the results or using real robots to run your algorithm on and also, get your results. Each method has its advantages and disadvantages, some advantages of simulators are you can deploy any number of robots as you wish, you can have crashes without harming any hardware and you get good results very near to real results but the disadvantage is the results are not as exact copy of the results you get from experimenting on the real robots as something can't be interpreted in the experiments like friction. As for real robots, the advantage is you get accurate results but the disadvantage is you can't deploy as much robots as you can, as robots are expensive to get leading to poor experimentations.

In this thesis, we will use a simulator to conduct our experiments. So, two major comparisons will be presented. The first comparison is a comparison between different robots available now, sure not all of the available robots, we chose four of them that are mostly used in the literature. The four robots are AntBot, ePuck, Swarmanoid and finally Kilobot. The second comparison will be related to the simulators to run the experiments on. The famous simulators nowadays for robotic swarm are v-rep, Webot and Argos and we will present a comparison between all the three.

A.1 Robotic Platforms

A.1.1 AntBot

AntBot is based on Pololu 3pi Robot with the addition to its expansion kit to increase the power and functionality of the simple robot.

- Locomotion
  - Two wheels that are accommodated with two direct current motors.
- Communication and Sensing

  - 5 IR emitter and receiver (phototransistor) pairs facing the ground.
  - MMA7455 3 axes accelerometer for collision detection.
  - Wixel Programmable USB wireless module "2.4GHz radio" used in communication between AntBots or between Antbots and PC.

- Controller
  - ARM Cortex M3 with 96 MHz and 32KB RAM and 512 KB FLASH (SPI, I2C, UART, PWM, ADC and GPIO) is used as the main control unit.
  - ATmega 328P is used to control the motors and IR sensors plus other components on the 3pi robot.

- Power System
  - AntBot is equipped with 4 AAA Ni-MH rechargeable batteries.

- Cost
  - AntBot cost around 250 USD per robot.

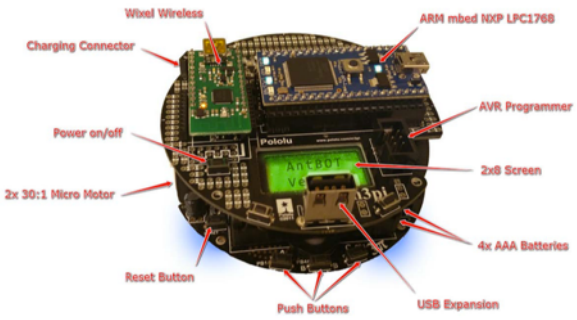

FIGURE A.1: AntBot Robot.

A.1.2 ePuck

- Locomotion
  - All wheels pf ePuck robot shown in Figure A.2 adopted from Cianci, Raemy, Pugh, *et al.* [29] are accommodated with stepper motors.

- Communication and Sensing
  - An extension connector to allow the addition of new functionalities on an extension board.
  - RS232 and Bluetooth interface to communicate with a host computer.
  - A 16 positions rotating switch allows an input from the user, for instance to set a running mode.
  - Reset button and programming connector as usual.

  - 8 IR proximity sensors are placed around the robot but not in a regular way (more sensors in front of the robot).
  - An accelerometer with 3 axes.
  - Three microphones and a speaker for sound capturing and generation.
  - A 640x480 color pixels resolution camera is placed inside the body and looks forward.
  - Can be used only through windowing or subsampling.
  - 8 red LEDs are placed around the robot body to display patterns. The camera should be able to see the other robots with their patterns.

- Controller
  - The e-puck is using a Microchip dsPIC microcontroller. This microcontroller complies with the educational criteria of flexibility because:
    * It is a 16-bit processor with a 16-entry register file.
    * Have a digital signal processor (DSP) unit.
    * It runs at 64 MHz and provides 16 MIPS of peak processing power.
    * The instruction set is mostly orthogonal 8 and rich; in particular, it contains multiply-accumulate and hardware-repeat instructions suitable to drive the DSP unit, for instance to efficiently compute scalar products and fast Fourier transforms.
    * Finally, it is supported by a custom tailored version of the GCC C compiler.
    * This version has an 8 KB of RAM and 144 KB of flash memory.

- Power System
  - Single power supply. All electronics runs at 3.3V, except the camera needing an additional 1.8V. LiION technology based battery is used, has 5Wh capacity and is sufficient for about 2-3 hours of intensive use. The battery can be removed and recharged externally. A battery protection is implemented.

- Cost
  - E-puck costs around 854 U.S. dollars per robot.
    As seen in `http://www.gctronic.com/shop.php`

A.1.3 Swarmanoid

Swarmanoid robot consist of three different groups of robots as shown in Figure A.3 adopted from Dorigo, Floreano, Gambardella, *et al.* [30], the first group of robots is called the foot-bots, these robots are small autonomous robots that can move on both

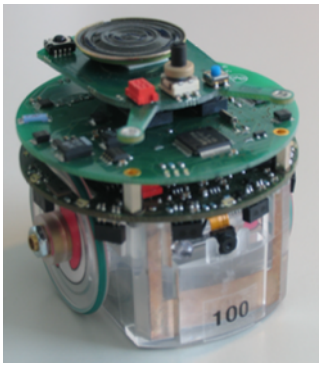

FIGURE A.2: ePuck Robot.

even and uneven terrains, capable of self-assembling and transporting objects. The second group of robots is called the hand-bots, these robots are small autonomous robots that can climb vertical surfaces and can manipulate objects. Finally, the third group of robots is called the eye-bots, these robots are small autonomous flying robots that can attach to the ceilings and analyze the environment to give information for the other two groups, the foot-bots and the hand-bots. Swarmanoid has no centralized control and relies on continued local and nonlocal interaction to produce a collective self-organized behavior.

- Locomotion
  - Foot-bots is 28 cm high and 13 cm diameter robot that has differential drive motion control that is composed of two 2-W motors each powers a rubber a rubber track and a wheel.
  - Hand-bots
    - No ground mobility but it can climb vertical structures and grasp small objects to bring them to the ground for the Foot-bots to transport them.
    - Magnet for attaching to ferromagnetic ceiling and a motor to switch the magnetic field.
    - Two fan propellers to provide the robot with orientation control.
    - Rope launcher.
    - 2 grasping hands
  - Eye-bots
    - Flying robot for indoors environment that can navigate through narrow corridors.
    - Ceiling attachment mechanism to scan the environment while disabling flying systems.
    - Navigate using the sensory information provided from other static eye-bots, this allows indoor navigation while avoiding the use of

GPS, 3D tracking camera, illumination dependent visual processing or expensive laser scanning.

* Eye-bots uses a quadrotor-like propulsion configuration but with a 4 X 2 coaxial rotor system.

- Communication and Sensing
  - Foot-bots
    * 24 IR sensors facing outwards for obstacle detection.
    * 12 IR sensors facing downwards for ground detection
    * Radio frequency identification reader and writer with antenna.
    * 3- Axis accelerometers and gyroscope for orientation measurements.
    * Gripper module that enables self-assembly between foot-bots and hand-bots this is done through a docking ring and a gripping mechanism.
    * 2-Dimensional force sensor to measure the effort applied on the dock ring.
    * RGB LEDs for color based communication.
    * 4 IR distance sensors.
    * Camera
    * LED beacon
    * WiFi
  - Hand-bots
    * High-resolution camera.
    * RGB LEDs for communicating with robots.
    * 4 IR distance sensors.
    * Color VGA camera and 12 distance sensor in each arm
    * WiFi
  - Eye-bots
    * Range and bearing communication device for coordinated movement in 3-D.
    * RGB LEDs for communicating with other robots.

- Controller
  - All robots use a distributed multiprocessor architecture where the main processor is responsible for intensive task such as vision and high level control and other microcontrollers are used for real-time sensing and controlling the actuators.
  - Main processor: 533-MHZ i.MX31 ARM11 with 128 MB of RAM and 64 MB of flash.

  - Microcontrollers: DsPIC 33 that has fixed point and DSP instructions

- Power System
  - Each robot is powered by lithium polymer battery with 3.7 V and 10 Ah.
  - A super capacitor is used to power the robot for 10 sec for battery exchange.

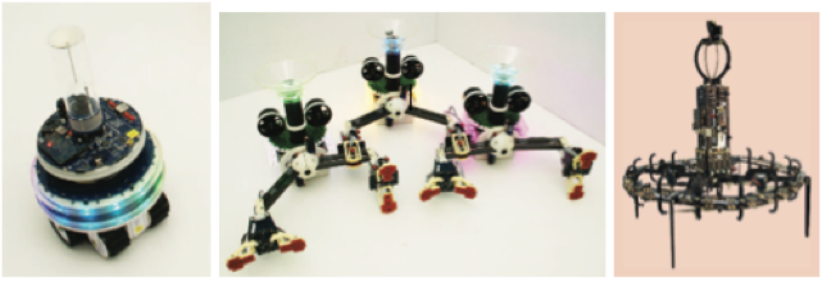

FIGURE A.3: Swarmanoid Robot. Left is the foot bot then the hand bot and finally the eye bot.

A.1.4 Kilobot

Design

- Locomotion
  - Kilobot as shown in Figure A.4 adopted from Rubenstein, Ahler, and Nagpal [31] uses two vibration motors for locomotion.
  - A forward force is generated from the centripetal forces generated from the motor vibrations.
  - Slip-stick principle principle explains the convertion of the motor vibration to a forward force.
  - Vertical rotation motion around the axis of motor is generated by activation of that motor.
  - By controlling the magnitude of vibration for the two motors independently in a differential drive manner, the robot can move in a continuous range from clockwise rotation, to straight forward, to counterclockwise rotation.
  - This enables the Kilobot to move approximately 1 cm/sec and rotate approximately 45 degrees/sec.

- Communication and Sensing
  - The sensing of neighbors only includes distance sensing, not bearing sensing.

  - In order to communicate with neighboring robots, each Kilobot has an infrared transmitter and receiver, which are located in the center of the PCB and are pointed directly downwards at the table the Kilobot is standing on. Both the transmitter and receiver have an isotropic emission or reception pattern. Which allow the robot to receive messages equally from all directions.
  - Additionally, both the receiver and transmitter are wide-angle, with an angle of half power of 60 degree from the robot's downward pointing vertical axis. When the transmitter is active, any nearby robot can receive the light emitted by the transmitting robot after it is reflected off the table,
  - Kilobot can communicate at rates up to 30 kb/s with robots up to 10cm (about 6 robot radii) away.
  - There is also a visible light sensor on each robot, which can sense the level of ambient light shining on the robot. That may be useful for other collective applications such as collective transport.

- Controller
  - A 32K memory Atmega328 microprocessor is used, which runs at 8 Mhz. This microprocessor has:
    * Two pulse width modulation (PWM) channels used for controlling the vibrating motors.
    * 10-bit ADC used for measuring the incoming infrared light intensity.
    * Self-programmable memory used to update the robot's program.
    * Low-power sleep mode.
  - C language is used for programming kilobot.

- Power System
  - A 3.4V/160 mAh lithium-ion battery is used.
  - The robot can run for 3 to 24 hours depending on the robot's activity level.
  - 30 µA is only drawn while in low-power states.
  - The charging will automatically stop once it is fully charged.

- Limitations
  - There is no real form of odometry. This makes moving precisely over long distances or for a long time difficult.
  - Another limitation to this locomotion is that it cannot move over rougher surfaces, requiring a smooth surface such as a dry erase surface to work.

- Cost
  - The Kilobot design uses about $14 worth of part.
  - This cost does not include the assembly of components on the PCB

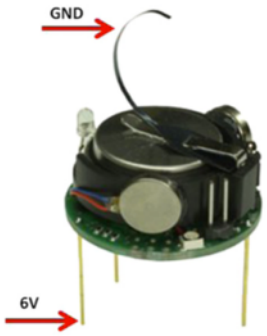

FIGURE A.4: Kilobot Robot.

Scalable Collection

An overhead infrared controller is used to allow the applying of scalable operation on all robots at the same time without the need for applying it on each robot individually. An operator sitting at a computer-based control station in turn controls the overhead controller.

- Power System
  - A low-power sleep mode is used instead of standard off state "power is switched off", in this mode the entire components are switched off except microprocessor is in sleep mode.
  - Every one minute the microprocessor wakes up to check wakeup signal from the controller, if a wakeup is received the robot is switched back on.
- Charging
  - Charging process occurs by placing the entire swarm over a conductive surface that will be connected to 6vdc then place a conducting board, on top of the swarm which act as the ground as shown in Figure A.5 adopted from Rubenstein, Ahler, and Nagpal [31].
- Programing
  - In this bootloader sector of memory, Kilobot has a program that receives infrared messages from the overhead controller, which contain portions of the new desired main program code. It then writes these portions of code to the appropriate location in the primary sector of memory.

After exploring all famous four robots in the literature and knowing their capabilities, we chose ePuck as the robot to use in our experiments as it has the physical capabilities we need to transport an object either by pushing or whatever technique we decide. Also, it has variety of ways of communication to maintain a channel between the robot and other robots in the environment.

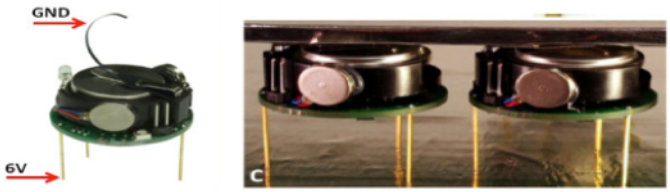

FIGURE A.5: Kilobot Robot Charging.

A.2 Simulation Platforms

A.2.1 VREP

V-rep is considered the Swiss army knife of all the famous simulators available in literature due to all the features, function and APIs that it provides. These features as mentioned on the simulator's website Robotics [32] and include

- Cross platform (Windows, Mac and Linux).
- 6 approaches for programming (Embedded scripts, plugins, add-ons, ROS nodes, remote API clients or custom solutions).
- 7 programming languages (C/C++, Python, Java, Lua, Matlab, Octave and Urbi).
- 400+ API functions.
- 100 ROS services, 30 publishers, 25 subscribers and extendable.
- 4 physic engines (ODE, Bullet, Vortex and Newton).
- POV-Ray integrated tracer.
- Full kinematics solver (IK and FK for any mechanisms).
- Mesh, octree and point cloud interference detection.
- Mesh, octree and point cloud minimum distance calculation.
- Built-in custom UI.
- Path/motion planning (holonomic in 2-6 dimensions, non-holonomic for car-like vehicles and motion planning for kinematic chains).
- Data recording and visualization (time graphs, X/Y graph or 3D curves)
- Model browse with drag/drop functionality (also during simulation)
- Realistic proximity sensors (minimum distance calculation within a detection volume)
- Integrated shape edits modes.

- Built-in image processing (fully extendable).
- Multi-level undo/redo, movie recorder, simulation of painy dispensing, exhaustive documentation, etc.
- Surface cutting simulation.
- Fully integrated reflexes motion library type 2 and RRS-1 interface specifications.

V-rep is considered an extremely customizable simulator. Every detail in this simulator is customizable, even the simulator itself can be personalized and customized to perform as needed. This can be achieved through the use of the Application Programming Interface (API). As mentioned above, V-rep provides six different approaches for coding or programming, these six approaches can be used all at the same time or mix and match between them to get the best functionality needed. The model, scene or simulator's control entity can be situated inside any of

- Embedded script: this is the easiest approach and mostly used. It is used in writing the Lua scripts. This approach can flexibly customize the scene, simulation and even the simulator.
- Add-on: this is used to customize the simulator itself, also, used in writing Lua scripts, the add-on can auto run or be called as a function and is supposed to be generic, i.e. not designed for a specific model or scene but the functionality should be bound to the simulator.
- Plugin:
- Remote API client: this allows the user to create functionalities on the robot or on another machine or use external application and still connect it to V-rep using API commands.
- ROS node: using the robot operating system, an external application can communicate with V-rep, this external application can be located on the robot or another machine.
- Custom client/server: this method needs to work with a plugin/script/communication mean (socket, pipes, etc). It is more flexible but need a lot of work to be done.

Communication and messaging mechanisms in V-rep is performed through different methods as shown in A.6 adopted from Rohmer, Singh, and Freese [33].

1. API calls from the main client application, the APIs can be either C/C++ or non-C/C++ application if the language supports calling to C functions.
2. Cascaded execution of child scripts.

3. Main script/child script/callback script/customized script calling the Lua API to the regular API. All calls are guided to the V-rep engine except for the custom functions written in Lua that call plugins.
4. Callback calls from the Lua scripts from the simulator to Lua custom function of the plugins.
5. Callback calls of events from simulator to plugins.
6. Remote APIs call from other robots, other machines or external applications.
7. Data exchange through ROS between V-rep and other robots, other machines or external applications.
8. Connections through sockets/pipes/... etc. to/from external applications.
9. An add-on calling the Lua API to the regular API. All calls are guided to the V-rep engine except for the custom functions written in Lua that call plugins.
10. Callback calls originated from V-rep to callback scripts.
11. Execution calls originated from V-rep to customized scripts.

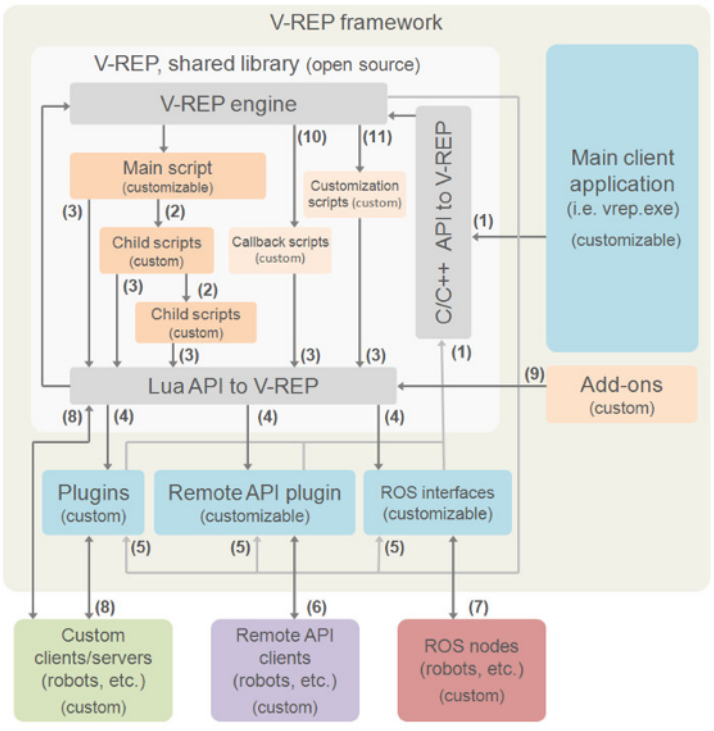

FIGURE A.6: V-rep communication and messaging mechanisms.

V-rep provides very powerful calculation methods that can be functions, modules. These methods are not directly connected to one entity (say sensor or actuator) but can be used to connect to more than one entity. These modules include

- Collision detection module

- Minimum distance calculation module
- Inverse kinematic calculation module
- Geometric constraint solver module
- Dynamics module
- Path and motion calculation module

A.2.2 WebotsTM

WebotsTM is made to be easy to use and very powerful simulator due to the presence of multiple implemented functions in it. These features as mentioned on the simulator's website CYBERBOTICS [34] and include

- Modeling and simulating any type of robots (legged, wheeled or flying).
- Sensors and actuators library.
- 200 API functions offered in 6 different programming languages (C/C++, Java, Python and Matlab), also 20 classes of APIs written in C/C++, Java or python and can be linked with external applications.
- Physics simulation using open dynamics engine.
- Simulation videos for web and public presentations.
- Available simulation examples with source code for commercial robots.
- Multi-agents simulation with facilities regarding local and global communication.
- Collision detection customized to the simulated object.
- Cross platform (Windows, Mac and Linux).
- Simulating infinite number of robots in the same scene.
- Interface for Matlab using the 200 functions of the Matlab API.
- ROS interface using either C++ or python programming languages, taking advantages of all ROS stacks like OpenCV, ... etc.
- Programming in supervisor's mode available only in Webots pro.
- Physics plugin programming available only in Webots pro.
- Built-in screenshots and video creation for visualizing the produced results.
- 3D graphics.

- Customizable source codes that can be integrated easily in the integrated source code editor.
- Scene editor with full-featured functionality.
- Easy design and integrate of robots created on CAD tools.
- Available libraries for available commercial robots.
- Indoor and outdoor libraries for objects.
- Inter robot communication.
- Wide range of sensors
- Simulating of real camera devices and process the images.
- LIDAR sensors (e.g. kinect, ... etc.)
- Display devices to display whatever needed from the robot's controller.
- Motors that can be customized as needed.
- Full ePuck robot with controller that is accurately calibrated.

Next let's discuss the available sensors and actuators in Webots simulator. In Webots, there is a fully integrated library for sensors and actuators to connect and calibrate on any robot. The sensors library includes

- Distance sensors.
- Range finders.
- Light sensors.
- Touch sensors.
- GPS (global positioning sensor).
- Cameras (1D, 2D, color and black and white).
- Compass.
- Position sensors for servos.
- Receivers.
- Inclinometers.

As for the actuators, the actuators library includes

- Servos.
- LEDs.

- Independent wheel motors.
- Differential wheel motors.
- Emitters.
- Grippers.

WebotsTM simulator enables fast simulators due to the use of virtual time in its simulation system. This helps the running of experiments much faster than running it on real robots. Subject to some criteria like the power of the computer used, along with the difficulty of the chosen setup for the experiment, the simulation might reach 300 times faster than running the experiments on real robot. Also, the simulator enables easy use of the graphical user interface while the simulation is running. Multiple options are provided by just a simple gesture like dragging the mouse. The position of the viewpoint can be changed, the orientation can be adjusted and just scrolling the mouse wheel can control the zoom. Dragging the mouse along with pressing the shift key can help moving or rotating any object. All the mentioned options and others make it easier for interactive testing.

WebotsTM simulator provides easy creation of difficult environment for experiments and testing on mobile robots. Using advanced hardware accelerated OpenGL technologies that involve lighting; mapping of texture, applying smooth shading, ... etc can help with environment creation. Using the VRML97 standard, WebotsTM simulator permits the importing of 3D models in the simulation scene from almost all the 3D modeling software. WebotsTM simulator provides optimization to any created world without taking into consideration its size, build as large worlds as possible and still Webots simulator will offer optimization for it. Webots simulator helps in building complex robots by connecting multiple chains of servo nodes; this means the creation of legged robots with multiple joints per leg, robotic arms, different camera systems, ... etc. as an example, placing and connecting multiple cameras helps in creating binocular stereovision or vision systems with 360-degree angle as mentioned in Michel [35].

A.2.3 ARGoS

ARGoS simulator is a famous simulator for being both efficient; due to the ability to simulate any number of robots and flexible; due to that the simulator can be highly customized and also, experiments can be extremely customized. ARGoS simulator uses an extremely modular approach. This modularity can be shown in A.7 adopted from ARGoS [36], the reasons behind this modularity are all aspects of simulations can be overridden to provide maximum flexibility, all modules are implemented as plugins that when simulating get loaded at runtime, ARGoS simulators permit the addition of multiple functionalities like adding new sensors, adding new actuators, adding visualizations, adding robots components, adding communication methods

and adding new physics engines, ARGoS simulator proposes a specific plugin called the loop functions that help in short lived simulation or specific extensions of experiments and in ARGoS simulator exists an API for the controlling of robots. This API is called the control interface and also exists on the real robot. This API helps the smooth transition between simulation and real robots.

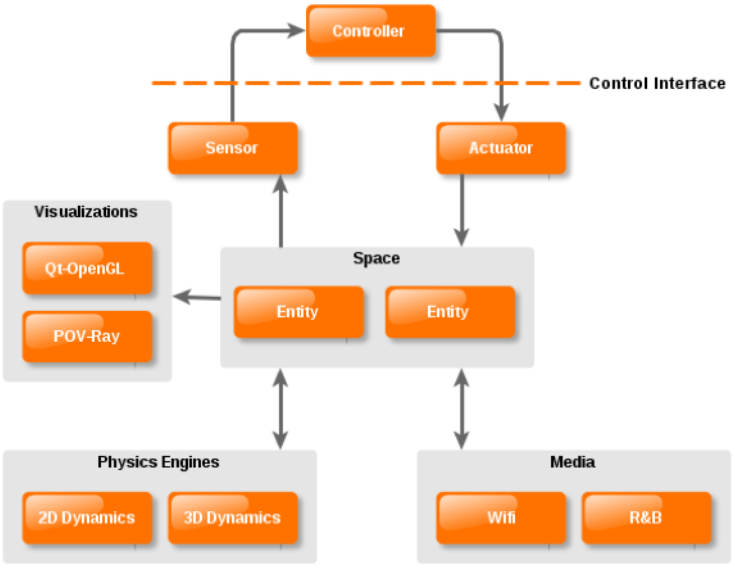

FIGURE A.7: ARGoS Modularity.

ARGoS provides multiple engines during simulation, and offers seamless transition between these engines. The robots are deployed in the simulation environment and starts moving around, along this movement, the robots shift between these engines automatically. The physics engines provided by the ARGoS simulator is just a plugin, the user freely chooses which physics engine to employ with the robots. Also, the physical simulation space is sub divided into multiple regions, each region corresponds to a physics engine that controls it. Of course, not only physics engines that distinguish the ARGoS simulator, communication is as important as physics engines. ARGoS simulator presented the idea of a medium. The medium is the name of a plugin implemented in ARGoS simulator that deals with all aspects of communication during simulations, i.e. range and bearing, wifi, ... etc. this medium is considered another type of engines that accomplish detailed simulations.

The ARGoS simulator supports the concept of multi core processing, where the simulation is subdivided between different master and slaves threads. In this concept, the master thread delegate to the slave thread the task to perform. Each task corresponds to a single plugin update (a sensor, an actuator, a component and an engine). The thread control is done while configuring the experiment. In ARGoS simulator, the work distribution among threads is performed by one of two methods, the scatter-gather method or the H-dispatch method. The scatter-gather method is shown in A.8 adopted from Pinciroli, Trianni, O'Grady, *et al.* [37], this method best

suits the tasks with the same computational cost, i.e. conducting an experiment of large number of similar robots. As for the H-dispatch method, it best suits the tasks with diverse computational cost, i.e. conducting an experiment of large number of different robots. Parallelism concept is fully integrated in the ARGoS simulator, so the developer is not concerned with the shared resources problem. Also, this is transparent for the developer of the plugins.

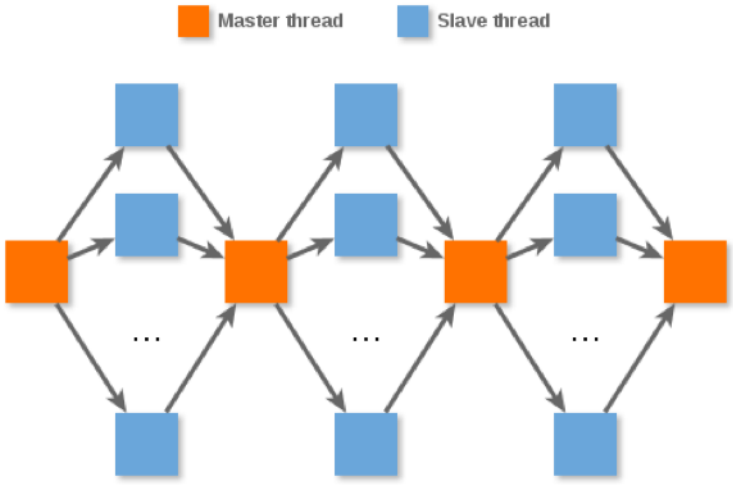

FIGURE A.8: ARGoS Parallelism.

Finalizing the comparison between the three simulators, going forward with our work, we will be using v-rep simulator to conduct all experiments. This decision was taken due to that the v-rep simulator is offering flexible simulator usage, very easy environments implementation and manipulation and is offering the connection with multiple APIs along with the help of plenty of programming languages. In addition, v-rep simulator is open source software hence free to use and program software, unlike WebotsRM simulator.

Appendix B

MRDPSO Algorithm

Appendix B will contain a brief description of our implemented algorithm along with its pseudo code.

B.1 MRDPSO Algorithm Pseudo Code

In this section, we discuss briefly our implemented algorithm as we already described it in Chapter 4. The MRDPSO algorithm consists of twelve stages that the moving robots move between them. All the robots start at the explore state and proceed according to what surround them in the environment. They move with the aim of finding the target that needs to be transported. The end of our algorithm is reached when all the targets in the environment are transported. The pseudo code for our algorithm is represented in Algorithm 3.

Algorithm 3 MRDPSO Algorithm

```
1:  procedure EPUCK_MAIN
2:      curr_state ← 'explore'
3:      while simulation_is_running do
4:          if curr_state == 'explore' then
5:              object_found ← move_random()
6:              if object_found == 'object' then
7:                  curr_state = 'obstacle'
8:              else if object_found == 'target' then
9:                  curr_state = 'target'
10:             end if
11:             if recruit_signal then
12:                 curr_state = 'reorientrobots'
13:             end if
14:         else if curr_state == 'obstacle' then
15:             // Obstacle avoidance technique
16:         else if curr_state = 'target' then
17:             target_corners_size ← 0
18:             number_needed_robots ← 1
19:             number_recruited_robots ← 1
20:             max_number_robots ← 0
21:             curr_state = 'sizeestimate'
```

Algorithm 3 MRDPSO algorithm (continued)

```
22:     else if curr_state == 'sizeestimate' then
23:         if target_corners_size <= 4 then
24:             target_corners ← target_rotation()
25:         else
26:             calculate_area()
27:             calculate_needed_robots()
28:             curr_state = 'recruit'
29:         end if
30:     else if curr_state == 'recruit' then
31:         while number_recruited_robots <= max_number_robots do
32:             send_recruit_signal()
33:             wait_confirmation_signal()
34:             if confirmation_signal == number_recruited_robots then
35:                 curr_state = 'preparetransport'
36:             end if
37:         end while
38:     else if curr_state == 'waitingrobots' then
39:         calculate_recruiter_position()
40:         object_found ← move_to_position()
41:         if object_found == 'obstacle' then
42:             curr_state == 'obstacle'
43:         end if
44:     else if curr_state == 'orientation' then
45:         object_found ← check_prox_sen()
46:         if object_found == 'obstacle' then
47:             curr_state = 'obstacle'
48:         else
49:             orient_robots()
50:             if target_found then
51:                 curr_state = 'transport'
52:             end if
53:         end if
54:     else if curr_state == 'preparetransport' then
55:         calculate_positions_for_recruited()
56:         signal_confirmation ← send_position_signal
57:         if signal_confirmation == number_recruited_robots then
58:             curr_state == 'waitingrobots'
59:         end if
60:     else if curr_state == 'reorientrobots' then
61:         wait_position_signal()
62:         if send_position_signal then
63:             send_confirmation_signal()
64:             curr_state = 'orientation'
65:         end if
66:     else if curr_state == 'transport' then
67:         if curr_position_y <= end_position_y then
68:             send_transported_signal()
69:             curr_state = 'donetransport'
70:         else
```

Algorithm 3 MRDPSO algorithm (continued)

```
                push_target()
            end if
        else if curr_state == 'orienttransport' then
            if transported_signal then
                signal_number ← count_signals()
                if signal_number == number_of_recruited_robots then
                    curr_state = 'donetransport'
                    send_target_transported_signal()
                end if
            end if
        else if curr_state == 'donetransport' then
            if all_targets_transported then
                stop_robots_movement()
                stop_simulation()
            else
                curr_state = 'explore'
            end if
        end if
    end while
end procedure
```

Bibliography

[1] A. Ghosh, A. Konar, and R Janarthanan, "Multi-robot cooperative box-pushing problem using multi-objective particle swarm optimization technique", in *Information and communication technologies (wict), 2012 world congress on*, IEEE, 2012, pp. 272–277.

[2] S. TORABI, "Collective transportation of objects by a swarm of robots", PhD thesis, Ms. Thesis, Chalmers University of Technology, 2015.

[3] C. R. Kube and E. Bonabeau, "Cooperative transport by ants and robots", *Robotics and autonomous systems*, vol. 30, no. 1-2, pp. 85–101, 2000.

[4] I. Navarro and F. Matía, "An introduction to swarm robotics", *Isrn robotics*, vol. 2013, 2012.

[5] Y. Hasan, "Swarms robots and their applications",

[6] T. J. Czaczkes and F. L. Ratnieks, "Cooperative transport in ants (hymenoptera: Formicidae) and elsewhere", *Myrmecol. news*, vol. 18, pp. 1–11, 2013.

[7] Y. Tan and Z.-y. Zheng, "Research advance in swarm robotics", *Defence technology*, vol. 9, no. 1, pp. 18–39, 2013.

[8] T. D. Seeley, P. K. Visscher, and K. M. Passino, "Group decision making in honey bee swarms", *American scientist*, vol. 94, no. 3, p. 220, 2006.

[9] V. Trianni, R. Groß, T. H. Labella, E. Sahin, and M. Dorigo, "Evolving aggregation behaviors in a swarm of robots", in *Ecal*, Springer, 2003, pp. 865–874.

[10] M. Brambilla, E. Ferrante, M. Birattari, and M. Dorigo, "Swarm intell swarm robotics: A review from the swarm engineering perspective", *Swarm intelligence*, vol. 7, no. 1, pp. 1–41, 2013.

[11] E. Sahin, S. Girgin, L. Bayindir, and A. E. Turgut, "Swarm robotics.", *Swarm intelligence*, vol. 1, pp. 87–100, 2008.

[12] Y. Mohan and S. Ponnambalam, "An extensive review of research in swarm robotics", in *Nature & biologically inspired computing, 2009. nabic 2009. world congress on*, IEEE, 2009, pp. 140–145.

[13] J. C. Barca and Y. A. Sekercioglu, "Swarm robotics reviewed", *Robotica*, vol. 31, no. 3, pp. 345–359, 2013.

[14] H. F. McCreery and M. Breed, "Cooperative transport in ants: A review of proximate mechanisms", *Insectes sociaux*, vol. 61, no. 2, pp. 99–110, 2014.

[15] R. Planqué, J. B. Van Den Berg, and N. R. Franks, "Recruitment strategies and colony size in ants", *Plos one*, vol. 5, no. 8, e11664, 2010.

[16] G. A. Pereira, M. F. Campos, and V. Kumar, "Decentralized algorithms for multi-robot manipulation via caging", *The international journal of robotics research*, vol. 23, no. 7-8, pp. 783–795, 2004.

[17] R. Groß, E. Tuci, M. Dorigo, M. Bonani, and F. Mondada, "Object transport by modular robots that self-assemble", in *Robotics and automation, 2006. icra 2006. proceedings 2006 ieee international conference on*, IEEE, 2006, pp. 2558–2564.

[18] J. Fink, M. A. Hsieh, and V. Kumar, "Multi-robot manipulation via caging in environments with obstacles", in *Robotics and automation, 2008. icra 2008. ieee international conference on*, IEEE, 2008, pp. 1471–1476.

[19] R. Gross and M. Dorigo, "Towards group transport by swarms of robots", *International journal of bio-inspired computation*, vol. 1, no. 1-2, pp. 1–13, 2009.

[20] M. Rubenstein, A. Cabrera, J. Werfel, G. Habibi, J. McLurkin, and R. Nagpal, "Collective transport of complex objects by simple robots: Theory and experiments", in *Proceedings of the 2013 international conference on autonomous agents and multi-agent systems*, International Foundation for Autonomous Agents and Multiagent Systems, 2013, pp. 47–54.

[21] K. Sugawara, N. Correll, and D. Reishus, "Object transportation by granular convection using swarm robots", in *Distributed autonomous robotic systems*, Springer, 2014, pp. 135–147.

[22] S. Wilson, T. P. Pavlic, G. P. Kumar, A. Buffin, S. C. Pratt, and S. Berman, "Design of ant-inspired stochastic control policies for collective transport by robotic swarms", *Swarm intelligence*, vol. 8, no. 4, pp. 303–327, 2014.

[23] T. Yu, T. Yasuda, K. Ohkura, Y. Matsumura, and M. Goka, "Apply incremental evolution with cma-neuroes controller for a robust swarm robotics system", in *Sice annual conference (sice), 2014 proceedings of the*, IEEE, 2014, pp. 295–300.

[24] G. Habibi, Z. Kingston, W. Xie, M. Jellins, and J. McLurkin, "Distributed centroid estimation and motion controllers for collective transport by multi-robot systems", in *Robotics and automation (icra), 2015 ieee international conference on*, IEEE, 2015, pp. 1282–1288.

[25] J. Chen, M. Gauci, W. Li, A. Kolling, and R. Groß, "Occlusion-based cooperative transport with a swarm of miniature mobile robots", *Ieee transactions on robotics*, vol. 31, no. 2, pp. 307–321, 2015.

[26] G. Habibi, W. Xie, M. Jellins, and J. McLurkin, "Distributed path planning for collective transport using homogeneous multi-robot systems", in *Distributed autonomous robotic systems*, Springer, 2016, pp. 151–164.

[27] E. Feo Flushing, L. M. Gambardella, and G. A. Di Caro, "On decentralized coordination for spatial task allocation and scheduling in heterogeneous teams", in *Proceedings of the 2016 international conference on autonomous agents & multiagent systems*, International Foundation for Autonomous Agents and Multiagent Systems, 2016, pp. 988–996.

[28] M. S. Couceiro, P. A. Vargas, R. P. Rocha, and N. M. Ferreira, "Benchmark of swarm robotics distributed techniques in a search task", *Robotics and autonomous systems*, vol. 62, no. 2, pp. 200–213, 2014.

[29] C. M. Cianci, X. Raemy, J. Pugh, and A. Martinoli, "Communication in a swarm of miniature robots: The e-puck as an educational tool for swarm robotics", *Lecture notes in computer science*, vol. 4433, pp. 103–115, 2007.

[30] M. Dorigo, D. Floreano, L. M. Gambardella, F. Mondada, S. Nolfi, T. Baaboura, M. Birattari, M. Bonani, M. Brambilla, A. Brutschy, *et al.*, "Swarmanoid: A novel concept for the study of heterogeneous robotic swarms", *Ieee robotics & automation magazine*, vol. 20, no. 4, pp. 60–71, 2013.

[31] M. Rubenstein, C. Ahler, and R. Nagpal, "Kilobot: A low cost scalable robot system for collective behaviors", in *Robotics and automation (icra), 2012 ieee international conference on*, IEEE, 2012, pp. 3293–3298.

[32] C. Robotics, *V-rep: Virtual robot experimentation platform*, 2015.

[33] E. Rohmer, S. P. Singh, and M. Freese, "V-rep: A versatile and scalable robot simulation framework", in *Intelligent robots and systems (iros), 2013 ieee/rsj international conference on*, IEEE, 2013, pp. 1321–1326.

[34] L. CYBERBOTICS, *Webots simulator*, 2015.

[35] O. Michel, "Cyberbotics ltd. webots™: Professional mobile robot simulation", *International journal of advanced robotic systems*, vol. 1, no. 1, p. 5, 2004.

[36] ARGoS, *Argos simulator*, 2015.

[37] C. Pinciroli, V. Trianni, R. O'Grady, G. Pini, A. Brutschy, M. Brambilla, N. Mathews, E. Ferrante, G. Di Caro, F. Ducatelle, *et al.*, "Argos: A modular, multi-engine simulator for heterogeneous swarm robotics", in *Intelligent robots and systems (iros), 2011 ieee/rsj international conference on*, IEEE, 2011, pp. 5027–5034.