

**MODELLING OF MULTI-ROBOT SYSTEM
FOR SEARCH AND RESCUE**

POY YI LER


**A project report submitted in partial fulfilment of the
requirements for the award of Bachelor of Mechatronics Engineering
with Honours**

**Lee Kong Chian Faculty of Engineering and Science
Universiti Tunku Abdul Rahman**

May 2023

DECLARATION

I hereby declare that this project report is based on my original work except for citations and quotations which have been duly acknowledged. I also declare that it has not been previously and concurrently submitted for any other degree or award at UTAR or other institutions.

Signature :  _____

Name : POY YI LER _____

ID No. : 1904296 _____

Date : 2 May 2023 _____

APPROVAL FOR SUBMISSION

I certify that this project report entitled **MODELLING OF MULTI-ROBOT SYSTEM FOR SEARCH AND RESCUE** was prepared by **POY YI LER** has met the required standard for submission in partial fulfilment of the requirements for the award of Bachelor of Mechatronics Engineering with Honours at Universiti Tunku Abdul Rahman.

Approved by,

Signature : 

Supervisor : Dr Shalini Darmaraju

Date : 20 May 2023

Signature : 

Co-Supervisor : Dr Kwan Ban Hoe

Date : 22 May 2023

The copyright of this report belongs to the author under the terms of the copyright Act 1987 as qualified by Intellectual Property Policy of Universiti Tunku Abdul Rahman. Due acknowledgement shall always be made of the use of any material contained in, or derived from, this report.

© 2023, Poy Yi Ler. All right reserved.

ACKNOWLEDGEMENTS

I would like to thank everyone who had contributed to the successful completion of this project. I would like to express my gratitude to my research supervisor and co-supervisor, Dr. Shalini A/P Darmaraju and Dr. Kwan Ban-Hoe for their invaluable advice, guidance and their enormous patience throughout the development of the research. Their expertise and commitment have been instrumental in ensuring the success of this project. It was a great privilege and honour to work and study under their guidance.

In addition, I would also like to express my gratitude to my loving parents and friends who have helped and given me encouragement and support throughout the whole journey of conducting this research. and providing valuable feedback, which helped me to refine my ideas and improve the quality of my work. Once again, thank you to everyone who contributed to the completion of this project.

ABSTRACT

The field of robotics has seen an increased interest in multi-robot systems, which bring a new set of challenges to the table. One of the key aspects in multi-robot systems is the path planning problem, which involves finding collision-free paths for each robot to reach their respective destinations while optimizing various performance metrics. This report focusses on developing a novel multi-robot path planning algorithm based on the Modified Particles Swarm Optimization (MPSO) algorithm for dynamic environments. The MPSO algorithm introduces a new path planning scheme for determining robot's waypoints. Unlike the normal PSO algorithm which initializes the particle swarm at the robot's starting position and iteratively determining each waypoint until a completed path is generated, MPSO algorithm initializes the particle swarm within a predefined search space and searches for the global best position within it to determine a specific robot waypoint through iteration updates. Moreover, to cope with dynamic environments, a combination of global and local path planning methods is introduced. The PSO algorithm functions as a global path planner, determining the complete path for each robot, whereas a sensor-based obstacle avoidance algorithm serves as a local planner to avoid collision with dynamic obstacles during navigation. In this project, this sensor-based algorithm is known as the Obstacle Avoidance Algorithm. The simulations conducted using MATLAB demonstrate the superiority of the MPSO algorithm over the PSO algorithm in terms of average path length and execution time of all robots in all three proposed scenarios: 16 meter shorter and 7.1 seconds faster in the first scenario, 17.89 meters shorter and 6.14 seconds faster in scenario 2, and 6.18 meters shorter and 8.47 seconds faster in scenario 3. The impact of the MPSO parameters on the simulation results is also studied to determine the best PSO parameters that achieve the best performance. It was found that the number of populations set to 75 and dynamically adjusts the value of inertial weight, the cognitive and social parameter provides the best performance in terms of shortest path length and execution time. In conclusion, this project shows that the MPSO algorithm is capable of generating a better path compared to the normal PSO algorithm in terms of average path length and execution time, making it a promising algorithm for multi-robot path planning in dynamic environments.

TABLE OF CONTENTS

DECLARATION	i
APPROVAL FOR SUBMISSION	ii
ACKNOWLEDGEMENTS	iv
ABSTRACT	v
TABLE OF CONTENTS	vi
LIST OF TABLES	viii
LIST OF FIGURES	x
LIST OF SYMBOLS / ABBREVIATIONS	xiv
LIST OF APPENDICES	xv

CHAPTER

1	INTRODUCTION	1
	1.1 General Introduction	1
	1.1.1 Search and Rescue (SAR)	2
	1.1.2 Mobile Robot	2
	1.1.3 Multi-robot System (MRS)	3
	1.2 Importance of the Study	4
	1.3 Problem Statement	6
	1.4 Aim and Objectives	7
	1.5 Scope and Limitation of the Study	7
	1.6 Contribution of the Study	8
	1.7 Outline of the Report	8
2	LITERATURE REVIEW	10
	2.1 Introduction	10
	2.2 Optimization Criteria	13
	2.3 Multi-robot Path Planning Algorithm	13
	2.3.1 Classical Method	14
	2.3.2 Heuristic Method	25
	2.4 Summary	39

3	METHODOLOGY AND WORK PLAN	41
3.1	Introduction	41
3.2	Proposed Multi-robot Path Planning Algorithm	41
3.2.1	Assumptions	41
3.2.2	Classical PSO Algorithm (PSO)	42
3.2.3	Modified PSO Algorithm (MPSO)	43
3.2.4	Dynamic Obstacle Avoidance	48
3.2.5	Flowchart of MPSO Path Planning Algorithm	50
3.2.6	Flowchart of Obstacle Avoidance Algorithm	51
3.3	Computer Simulations	52
3.4	Planning and Managing of Project Activities	54
3.4.1	Project Part I	54
3.4.2	Project Part II	55
3.4.3	Summary	58
4	RESULTS AND DISCUSSION	59
4.1	Introduction	59
4.2	Comparison Between PSO and MPSO Algorithms	59
4.2.1	Scenario 1 (Four Circular Obstacles)	61
4.2.2	Scenario 2 (Seven Circular Obstacles)	68
4.2.3	Scenario 3 (Multiple Long walls)	74
4.3	MPSO Parameters	82
4.3.1	Population Size	82
4.3.2	Inertial Weight	84
4.3.3	Cognitive and Social Learning Factor	85
4.4	Summary	87
5	CONCLUSIONS AND RECOMMENDATIONS	88
5.1	Conclusions	88
5.2	Recommendations for Future Work	89
	REFERENCES	91
	APPENDICES	98

LIST OF TABLES

Table 2.1:	Comparison of Computation Efficiency of ACO and GA (TAN, 2007)	28
Table 2.2:	Comparison of Path Length Generated Using ACO and GA (Liu, Mao and Yu, 2006)	29
Table 2.3:	Path Length Comparison of D2PSO and DPSO (Ayari and Bouamama, 2017)	33
Table 2.4:	Time Parameter for All the Nodes(Han, Zhou and Chen, 2016)	36
Table 2.5:	Schedule of All Robots in The Collision Region(Han, Zhou and Chen, 2016)	36
Table 2.6:	Number of Steps Taken to Reach the Goals(Das, Behera and Panigrahi, 2016)	38
Table 2.7:	Average Distance Travelled and Time Taken Using IPSO-IGSA, IPSO, and IGSA (Das, Behera and Panigrahi, 2016)	38
Table 2.8:	Comparisons between RRT, APF, ACO and PSO Algorithm	39
Table 3.1:	Gantt Chart (Part I)	56
Table 3.2:	Gantt Chart (Part II)	57
Table 4.1:	PSO Parameters	60
Table 4.2:	Starting and Goal Position for Each Robot	60
Table 4.3:	Global Best Fitness of Each Iteration in Scenario 1 With PSO	63
Table 4.4:	Global Best Fitness of Each Iteration in Scenario 1 With MPSO	66
Table 4.5:	Result Obtained with PSO and MPSO Algorithms in Scenario 1	67
Table 4.6:	Global Best Fitness of Each Iteration in Scenario 2 With PSO	69
Table 4.7:	Global Best Fitness of Each Iteration in Scenario 2 With MPSO	72

Table 4.8:	Result Obtained with PSO and MPSO Algorithms in Scenario 2	73
Table 4.9:	Global Best Fitness of Each Iteration in Scenario 3 With PSO	76
Table 4.10:	Global Best Fitness of Each Iteration in Scenario 3 With MPSO	79
Table 4.11:	Results Obtained with PSO and MPSO Algorithms in Scenario 3	80
Table 4.12:	Simulation results with different population sizes	83

LIST OF FIGURES

Figure 1.1:	MRS for Search and Rescue (Drew, 2021)	4
Figure 1.2:	Total number of publications per topic based on keyword search (Drew, 2021)	5
Figure 2.1:	Classification of multi-robot path planning (Lamini, Fathi and Benhlina, 2017)	11
Figure 2.2:	Classification of MRPP Based on Local and Global Environment (Zhang, Lin and Chen, 2018)	12
Figure 2.3:	Total Number of Paper Retrieved from the Database of Engineering Village (Zhang, Lin and Chen, 2018)	12
Figure 2.4:	Combination of Attractive Force and Repulsive Force (Matoui <i>et al.</i> , 2017)	15
Figure 2.5:	Illustration of Local Minimum Cases (Abdalla, Abed and Ahmed, 2017)	16
Figure 2.6:	Illustration of Path Oscillation Case (Jing Ren, McIsaac and Patel, 2006)	16
Figure 2.7:	(a) Simulation with APF Which Local Minimum Occurs, (b) Simulation with APF and SA Where SA Able to Help the Robot to Escape Local Minimum. (Zhu, Yan and Xing, 2006)	17
Figure 2.8:	Path Planning of the three Algorithms (Wu, Su and Li, 2019)	18
Figure 2.9:	Cooperation of three Robots with The Present of Static Obstacles (Matoui <i>et al.</i> , 2017)	19
Figure 2.10:	Illustration of RRT Path Planning Algorithm (Lee, Lee and Shim, 2017)	20
Figure 2.11:	(a) Comparison of RRT and (b) RRT* Algorithm (Bohács, Gyimesi and Rózsa, 2016)	22
Figure 2.12:	The Distance Between Robots using RRT* (Yang Li <i>et al.</i> , 2013)	22
Figure 2.13:	Path Generated Using RRT* for Two Mobile Robots (Yang Li <i>et al.</i> , 2013)	23

Figure 2.14:	Result of RRT (a) and RRT* (b) Using 5000 Nodes (Connell and Manh La, 2018)	24
Figure 2.15:	(a) Initial Position of Moving Obstacle. (b) Final Position of Moving Obstacles. (c) Result of RRT* Algorithm. (d) Final Obstacles Position and Executed Path (Connell and Manh La, 2018)	24
Figure 2.16:	Execution time of RRT and RRT* (Connell and Manh La, 2018)	25
Figure 2.17:	Illustration of ACO (Patle <i>et al.</i> , 2019)	27
Figure 2.18:	(a) Sub-optimal Path Using Dijkstra Algorithm, (b) Globally Optimal Path with ACO (TAN, 2007)	27
Figure 2.19:	Illustration of Dead Corner and Route Deadlock (Liu, Mao and Yu, 2006)	28
Figure 2.20:	(a) Environment I and (b) Environment II (Liu, Mao and Yu, 2006)	29
Figure 2.21:	Illustration of PSO (Iran Macedo, 2018)	31
Figure 2.22:	(a) In Start Time, (b) In Middle Time, (c) In End Time (Maryam Yarmohamadi and Hossein Erfani, 2011)	32
Figure 2.24:	Result Obtained using D ² PSO (Ayari and Bouamama, 2017)	33
Figure 2.25:	Searching Procedure of PSO (Biswas, Anavatti and Garratt, 2017)	34
Figure 2.26:	SRVPSO Path Planning with Two Robots with Obstacles Avoidance (Biswas, Anavatti and Garratt, 2017)	35
Figure 2.27:	(a) Simulation Result and (b) Project Diagram (Han, Zhou and Chen, 2016)	36
Figure 2.28:	Simulation Result of IPSO-IGSA Path Planning Using Six Robots (Das, Behera and Panigrahi, 2016)	38
Figure 3.1:	Infeasible Path When Obstacle Blocks Path	47
Figure 3.2:	Sensor Angle Coverage of Robot	49
Figure 3.3:	Flowchart PSO Path Planning Algorithm	50
Figure 3.4:	Flowchart for Obstacle Avoidance Algorithm	51

Figure 3.5:	Scenario 1 (Simulation Environment with four Circular obstacles)	52
Figure 3.6:	Scenario 2 (Simulation Environment with Seven Circular Obstacles)	53
Figure 3.7:	Scenario 3 (Simulation Environment with Multiple Vertical and Horizontal Obstacles)	53
Figure 4.1:	Particles' Position and Waypoints in Scenario 1 With PSO Algorithm: (a) Robot 1; (b) Robot 2; (c) Robot 3	61
Figure 4.2:	Trajectory of Robots in Scenario 1 With PSO Algorithm	62
Figure 4.3:	Dynamic Obstacle Detection of Robot 1 and Robot 3 in Scenario 1 (Simulation Step 5)	62
Figure 4.4:	Collision Avoidance Algorithm Triggered to Avoid Collision in Scenario 1 (Simulation Step 8)	63
Figure 4.5:	Graph of Global Best Fitness vs Iteration for Scenario 1 With PSO Algorithm	64
Figure 4.6:	Particles' Position and Waypoints in Scenario 1 With MPSO Algorithm: (a) Robot 1; (b) Robot 2; (c) Robot 3	65
Figure 4.7:	Trajectory of Robots in Scenario 1 With MPSO Algorithm	66
Figure 4.8:	Graph of Global Best Fitness vs Iteration for Scenario 1 With MPSO Algorithm	67
Figure 4.9:	Particles' Position and Waypoints in Scenario 2 With PSO Algorithm: (a) Robot 1; (b) Robot 2; (c) Robot 3	68
Figure 4.10:	Trajectory of Robots in Scenario 2 With PSO Algorithm	69
Figure 4.11:	Graph of Global Best Fitness vs Iteration for Scenario 2 With PSO Algorithm	70
Figure 4.12:	Particles' Position and Waypoints in Scenario 2 With MPSO Algorithm: (a) Robot 1; (b) Robot 2; (c) Robot	71
Figure 4.13:	Trajectory of Robots in Scenario 2 With MPSO Algorithm	72
Figure 4.14:	Graph of Global Best Fitness vs Iteration for Scenario 2 With MPSO Algorithm	73

Figure 4.15:	Particles' Position and Waypoints in Scenario 3 With PSO Algorithm: (a) Robot 1; (b) Robot 2; (c) Robot 3	74
Figure 4.16:	Trajectory of Robots in Scenario 3 With PSO Algorithm	75
Figure 4.17:	Dynamic Obstacle Detection of Robot 2 and Robot 3 in Scenario 3 (Simulation Step 6)	75
Figure 4.18:	Collision Avoidance Algorithm Triggered to Avoid Collision in Scenario 3 (Simulation Step 7)	76
Figure 4.19:	Graph of Global Best Fitness vs Iteration for Scenario 3 With PSO Algorithm	77
Figure 4.20:	Particles' Position and Waypoints in Scenario 3 With MPSO Algorithm: (a) Robot 1; (b) Robot 2; (c) Robot 3	78
Figure 4.21:	Trajectory of Robots in Scenario 3 With MPSO Algorithm	79
Figure 4.22:	Graph of Global Best Fitness vs Iteration for Scenario 3 With MPSO Algorithm	80
Figure 4.23:	Scenario 1 Used to Evaluate the Effect of PSO Parameters	82
Figure 4.24:	Simulation Results Obtained for Different Population Size	83
Figure 4.25:	Result Obtained with different learning factor for 30 iterations (Cognitive Learning Factor = 2, Social Learning Factor = 0)	86
Figure 4.26:	Result obtained for 30 iterations with cognitive learning factor = 0 and social learning factor = 2)	87

LIST OF SYMBOLS / ABBREVIATIONS

SAR	Search and Rescue
MRS	Multi-robot System
MRPP	Multi-robot Path Planning
APF	Artificial Potential Field
RRT	Rapidly Exploring Random Tree
GA	Genetic Algorithm
ACO	Ant Colony Optimization
PSO	Particle Swarm Optimization
MPSO	Modified Particle Swarm Optimization
$V_i(t + 1)$	Velocity of i^{th} particle for next iteration, $t+1$
$x_i(t + 1)$	Position of i^{th} particle for next iteration, $t+1$
$V_i(t)$	Current velocity of i^{th} particle at current iteration, t
$x_i(t)$	Current position of i^{th} particle at current iteration, t
x_{pBest_i}	Current personal best position of i^{th} particle
x_{GBest}	Current global best position achieved by the swarm.
ω	Inertial weight factor
c_1	Cognitive learning factor
c_2	Social learning factor
φ_1, φ_2	Independent variables uniformly distributed in $[0,1]$
K	Maximum number of iterations
k	Current iteration

LIST OF APPENDICES

Appendix A: Pseudocode	98
Appendix B: Simulations	100
Appendix C: Gantt Chart	106

CHAPTER 1

INTRODUCTION

1.1 General Introduction

Catastrophic events such as hurricanes, tsunamis, volcanic eruption, earthquake, storm, flood, and droughts are classified as natural disaster. Over the past 2 decades, the world has witnessed a significant number of natural disasters that have had a monumental impact, resulting in the loss of millions of lives globally (Prasad and Francescutti, 2017). For instance, the Indian Ocean tsunami happened in 2004 that claimed the lives of over 225, 000 people across a dozen countries (Britannica, 2022), Haiti earthquake in 2010 that caused approximately 220, 000 deaths (United Nation, 2022) and the serious Tohoku tsunami and earthquake happened in 2011 that claimed over 15, 500 lives (Becky Oskin, 2022). These examples are just a few among hundreds of natural disasters that occur every year, resulting in loss of lives and damage to economics. Despite the fact that natural disasters are a natural part of the planet's functioning, they are unavoidable.

In recent years, natural disasters have garnered much attention, and people are increasingly concerned about their frequency and severity. According to Intergovernmental Panel on Climate Change (IPCC), experts indicated that the global temperature is steadily increasing over the years, and it is significantly impacting the number and frequency of natural hazards and to some extent over the last decade, we have seen just that even in Malaysia (Zurich, 2022). Floods have occurred a lot more frequently and more intense in Malaysia over the past few years that resulting in loss of life and affecting more than hundred thousand citizens. It is expected that there will be more severe natural hazards such as storms, droughts, and wildfires in the future, and it is impossible to avoid them on our dynamic planet. With the world bracing for this increased rate of disaster events and to lower down the number of casualties caused by these natural disaster or natural hazards, search and rescue team plays a vital role in disaster recovery.

1.1.1 Search and Rescue (SAR)

During large-scale crises, such as Tohoku earthquake as mentioned previously, a well-designed and efficient search and rescue operations are crucial in saving lives. A simple SAR mission consists of a range of tasks which include searching, locating, rescuing and providing medical assistance to victims that are trapped inside a hazardous space (Doroftei, Matos and de Cubber, 2014). As such, SAR operations are of great importance in reducing casualties caused by disaster events. Given the life-threatening conditions faced by victims, including exposure to radiation, poisonous gases, and extreme temperature, it is vital for SAR teams to quickly find and rescue survivors, otherwise, the likelihood of finding victim alive will drops substantially (Siobhan Grayson, 2014). Traditional SAR operations rely heavily on human resources to perform these tasks, which can be challenging and demanding. The longer it takes to find and rescue victims, the lower their chances of survival. Therefore, reducing the time required for SAR operations is critical in minimizing casualties caused by natural disasters.

Besides that, SAR missions are often intricate, and disaster-prone environments pose a significant challenge as they are hazardous and challenging to navigate for human workers. Additionally, the crisis managers themselves are at risk of losing their lives in such environments. Consequently, robotics technology has been incorporated in SAR operations as a solution to these issues.

1.1.2 Mobile Robot

Robotics technology is increasingly being used in SAR operations due to the dangerous and complex nature of such missions. Unlike human rescuers, robots are able to bypass danger and can be deployed immediately to search for victims. Besides, robots are also capable of reaching areas that may be too dangerous or inaccessible for human workers, such as narrow spaces and unstable structure (Siobhan Grayson, 2014). At the same time, in addition to reducing the risk of human rescuers being exposed to hazardous environments, the use of robots can also increase the speed of response.

Implementing robotics in SAR operation presents both a challenging and promising solution to augment human rescuers in various ways and is a

fascinating area of research. Throughout the years, different types of robots have been employed in SAR operations across different environments such as on the ground, in the air, on the water surface, and underwater. For instance, drones have been utilized to search for missing people, providing situation awareness, create real-time maps, monitor, and analyse disasters (altigator, no date). To date, most rescue robots used are teleoperated, meaning that they are controlled by human operators from a distance. However, as the SAR operations become more intricate, a single teleoperated robot has a lot of drawbacks. A single robot result in a single point of failure, making it impossible to complete the mission if the robot is damaged, disabled, or trapped. Furthermore, a single robot often having complex design to navigate through all kind of terrain and perform all kind of tasks, resulting in the cost of one single robot being very high. In order to address this limitation, research has been extended by implementing multi-robot system into SAR operations with robots equipped with varying degrees of autonomy to minimize the workload of human operators in high-pressure disaster scenarios.

1.1.3 Multi-robot System (MRS)

Several factors make multi-robot system (MRS) particularly attractive in the context of SAR. The introduction of MRS results in faster responses and increased robustness due to the fact that extremely large areas can be effectively explored and each of the robot becomes dispensable. Besides searching for victims in needs, responses such as network infrastructure installation and map generation could also benefit from having multiple robots working together simultaneously (Drew, 2021). Furthermore, the issue of high cost of designing a single robot can be addressed. Instead of integrating all necessary hardware and capabilities on a single robot, it is more economical to distribute the necessary hardware such as sensor and actuators among multiple different robots (Queralta *et al.*, 2020). For instance, as shown in Figure 1.1, multiple robots can be used with each equipped or designed with different hardwares, some robots can be equipped with gripper to move obstacles, some robots can be equipped with camera to detect victims and some robots can be used to carry victims out from hazardous areas. Moreover, collaborative

efforts between different types of robots can also be leveraged to tackle complex tasks that are beyond the capabilities of a single robot.



Figure 1.1: MRS for Search and Rescue (Drew, 2021)

1.2 Importance of the Study

The current era of Industry 4.0 has led to significant advancements in robotics technology, opening up opportunities for automation in various industries including engineering, medicine, entertainment, and more. One crucial area of research in robotics is the study of MRS, which involves a group of robots working together to accomplish a task. In such a sense, a manufacturing line which consist of several robotics arm manipulators, mobiles robots and CNC machines can be considered as a MRS. In the context of SAR operations, the use of MRS has become increasingly important as natural disasters become more frequent and severe due to climate change. According to Petteri Taalas who is the Secretary-General of World Meteorological Organization (WMO), the increment in the number and severity of climate, water and weather extremes around the world are the result of climate change, and this also means that there will be more weather-related natural hazards in the future such as drought, forest fire, heatwaves and so on (United Nation, 2021).

The primary objective of SAR operations is to save lives as quickly as possible. MRS has high potential to significantly improve the efficiency of SAR personnel by improving response times, aiding in initial assessments, and mapping disaster areas. Besides that, MRS can reduce the risk of injury to rescuers by performing hazardous tasks. Currently, a single teleoperated robot is mostly used in SAR operation and the MRS are still not mature enough for widespread deployment in SAR operations (Drew, 2021; Chitikena, Sanfilippo and Ma, 2023). However, progresses are being made consistently over the past

two decades. Until today, researchers are still focusing on confronting the challenges of simulation-to-reality transfer and on realistic evaluation of constituent technologies in hope that the gap between the current capabilities and those required in a real disaster response can be reduced. According to Springer handbook of robotics, published field report data shows that there were a lot of documents related to robot-assisted response SAR operation, very few of them are involving three or more robots and none of the robots are fully autonomous (Bruno Siciliano and Oussama Khatib, 2007). Besides, from a graph presented in (Drew, 2021) as shown in Figure 1.2, the graph shows that total number of publications related to MRS used in search and rescue is far less than that of publications related to rescue robot alone. Therefore, while there are a lot of research done in rescue robot, the research on MRS is still lacking and successful cases of MRS deployment in SAR operations are scarce. Although MRS holds great promise for SAR, bridging the gap between academic interest and practical implementation in the field remains a significant challenge.

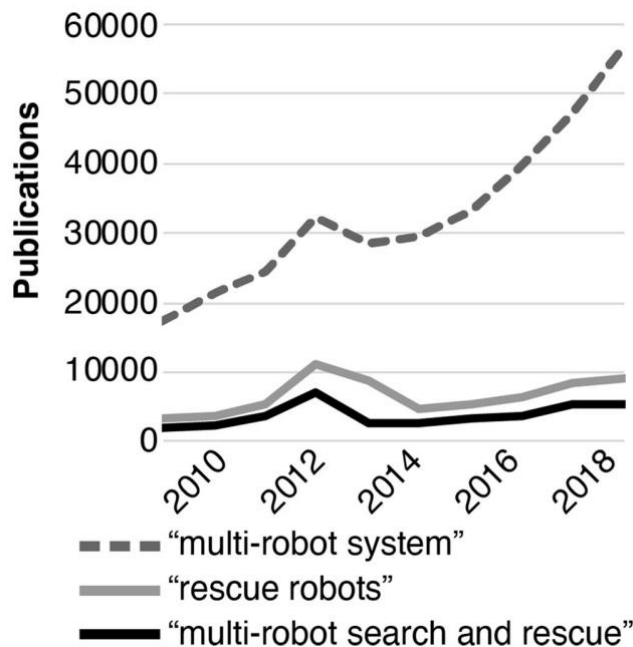


Figure 1.2: Total number of publications per topic based on keyword search (Drew, 2021)

1.3 Problem Statement

Despite the potential benefits of using MRS in SAR operations, it is a complex task to integrate them effectively. There are numerous factors to consider in designing an effective MRS, such as the architecture of MRS, centralized or decentralized; the ability of robots to recognize other robots; the task allocation for each of the robot; coalition formation; middleware support; the human-robot interaction; the collaboration among robots; the communication modalities and more (Gautam and Mohan, 2012).

Apart from that, the navigation of robots is also a crucial aspect that requires attention when implementing MRS, particularly in SAR operations. To successfully complete a navigation task, the robot must know its position in the environment and the position of its goal as well as the method for it to reach the goal position from initial position. The navigation of robot includes three primary functions which are localization, mapping, and path planning. However, in this research project, we will be concentrating solely on path planning. Path planning of a robot is just simply means to find the optimal path for the robot from its starting location to goal location. The complexity and robustness of the path planning algorithm dictate the time distance required for the robots to travel from starting location to the goal location which is where the victims located. As the victims might be trapped in a hazardous environment, the path planning algorithm need to be well designed so that the robots can reach its destination with the quickest route in the shortest time possible.

Humans are able to perform path planning effortlessly, such as avoiding obstacles that were not present before (Tzafestas, 2014). However, robots lack the same level of intelligence as humans because it needs to consider collision avoidance when planning their path from the starting location to the goal location. A good path planning algorithm takes into account both the static and dynamic obstacles to ensure that all robots can reach their destination quickly and without any damage. By incorporating collision avoidance into the path planning algorithm, robots can efficiently navigate through the environment while avoiding any obstacles.

In fact, MRS particularly multi-robot path planning (MRPP) is not a new concept and has been an interesting research topic over the past two

decades with numerous studies being conducted especially during this era of industrial 4.0. In the literature, there are a lot of research being done in designing a path planning algorithm and this number has continued to increase over the years. Therefore, it is a challenging task to identify which approach of designing a MRPP algorithm is appropriate for SAR operations as there are too many available approaches out there ranging from graph-based algorithm such as A* Algorithm to biologically inspired algorithm such as Genetic Algorithm (Debnath *et al.*, 2021). Moreover, even though path planning of MRS is relatively mature in highly structured environments, it remains difficult to construct an algorithm that works in situations similar to disaster areas, such as environments with dynamic obstacles or complex geometries, or in fluctuating environmental conditions like smoke. Not only that, building an algorithm that works in such dynamic and complex environments requires high computational speed and high response time, which poses another significant challenge.

1.4 Aim and Objectives

This project aims to explore the approaches and algorithms used for multi-robot path planning in search and rescue. By achieving the aim, the optimal path planning algorithm which able to generate shortest path with collision avoidance for each of the robots will be developed. The specific objectives of this project are listed as shown below:

- 1) To explore, evaluate and compare pros and cons of existing multi-robot path planning algorithms.
- 2) To develop an efficient MRPP algorithm to generate an optimal or shortest path with collision avoidance for each single robot between start position to destination position (position of survivors).
- 3) To demonstrate and evaluate the performance of the search and rescue mission through MATLAB simulation.

1.5 Scope and Limitation of the Study

As mentioned in the problem statement, constructing an MRS is a challenging task. This project's scope will be focusing on building a multi-robot point-to-point path planning algorithm and simulate the SAR operations using

MATLAB. In order to simplify the simulation and decrease the computational complexity, other aspects of MRS such as task allocation, collaboration among robots and communication between robot will be disregarded. At the same time, instead of heterogeneity system where different types of robots are implemented, a homogeneous system is to be designed where all the robots used in simulation are two-wheels robot. Besides, since this project is expected to be carried out for only within one year and to further simplify this project, the SAR simulation environment is assumed to be a known environment or partially known environment where the location of static obstacles, starting location of robots and location of victims are already known.

1.6 Contribution of the Study

This project outlines the existing path planning algorithm used to solve the multi-robot path planning problem. Each existing path planning algorithm is evaluated, and a path planning algorithm will be selected and developed with the aims to generate the shortest path with shortest execution time in a cluttered environment. The proposed algorithm was modified and fine-tuned to improve the final simulation results.

1.7 Outline of the Report

This report is divided into five chapters. In Chapter 1, the introduction of the whole project is discussed. The introduction consists of multiple sections such as the general introduction on search and rescue and application of multi-robot system in search and rescue mission, the importance of the study, problem statement, aims and objectives, scopes and limitation of the study, contribution of the study, and the outline of the whole report.

Chapter 2 provides a literature review of the project. The optimization criteria for solving a path planning problem are discussed. Besides that, the existing path planning methods and algorithms such as the classical and heuristic path planning algorithms are discussed and evaluated.

Chapter 3 outlines the methodology and work plan for the whole project. The methodology of the proposed MPSO algorithm is explained step by step. The mathematical formula of the proposed algorithm is also discussed

in this chapter to have a deeper understanding. Moreover, Gantt chart for the whole project is also included.

Chapter 4 presents the results and discussion. The result generated with the proposed MPSO algorithm and the PSO algorithm are evaluated in three different scenarios, and the performance for both the algorithms are compared to each other to determine the superior algorithm among the two algorithms. Additionally, the impact of the PSO parameters on the simulation result is also assessed in this chapter.

Chapter 5 discusses the conclusion of the report. The limitations of the whole project are discussed and the recommendation for future works are also included in this chapter.

CHAPTER 2

LITERATURE REVIEW

2.1 Introduction

Multi-robot path planning plays a vital role in the development of MRS as it enables robots to navigate from the starting location to the destination point in a given SAR mission with the shortest distance and minimum computation time, thereby conserving energy and minimizing potential hazards. In SAR, it is crucial for a path planning to produce the optimal path and should also hold the completeness criteria which ensure that at least one path can be found if that exists. However, there is still chances that the robot could not find a path to its destination due to all the paths being blocked by obstacles. In this case, the robot will never be able to reach the destination and can get stuck during the navigation.

As illustrated in Figure 2.1, the overall mobile robot path planning can be categorized into two categories which are the classical and heuristic methods (Patle *et al.*, 2019). The classical method was widely used before artificial intelligent techniques were developed. However, the major drawback of classical approach is that they do not perform well in dynamic environment or do not respond well to any uncertainties in the environment such as the behaviours of dynamic obstacles, weather condition and sudden changes of terrain due to collapsed building. Additionally, it incurs a high computational cost. In order to overcome this inefficiencies of classical method, heuristic method was developed. Heuristic method on the other hand has great ability in handling any uncertainties that present in the environment and thus more suitable to be used in dynamic environment and real time application (Patle *et al.*, 2019).

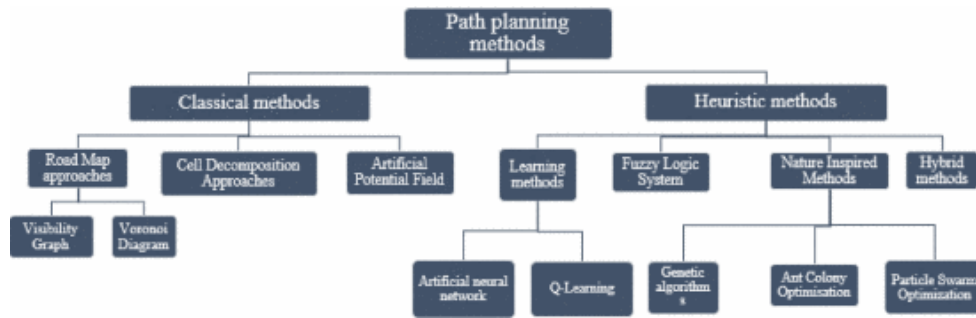


Figure 2.1: Classification of multi-robot path planning (Lamini, Fathi and Benhlma, 2017)

Besides, this MRPP problem can be solved using two techniques which are known as global and local path planning techniques. Some of the path planning methods or algorithms can be implemented in local environment and they are known as local path planning algorithm, some of them can be implemented in global environment which is known as global path planning algorithm. Furthermore, some algorithms can also be implemented in both local and global environment. Global path planning refers to the approach where the robots have prior knowledge of the environment before planning the optimal path, and it is also associated with offline path planning (Koubaa *et al.*, 2018). On the other hand, local path planning assumes that the robots lack the information about its environment, and they need to navigate in partially known or unknown environment. Consequently, the robots need to sense the location for obstacles in real time when searching for the optimal path to avoid clashing with the obstacles. Local path planning is associated with online path planning (de Almeida *et al.*, 2020). Some of the path planning approaches or algorithm for local and global environments are illustrated in Figure 2.2.

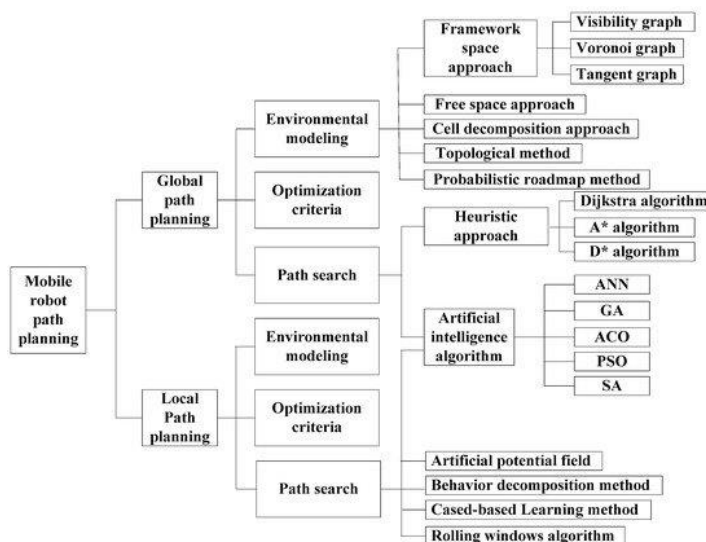


Figure 2.2: Classification of MRPP Based on Local and Global Environment (Zhang, Lin and Chen, 2018)

Other than that, in 2018, the number of papers related to mobile robot path planning retrieved using the Engineering Village database is shown in Figure 2.3 (Zhang, Lin and Chen, 2018). From the figure, we can observe that a lot of research have been done on path planning using various type of algorithms and Genetic Algorithm is the most popular one used in 2018.

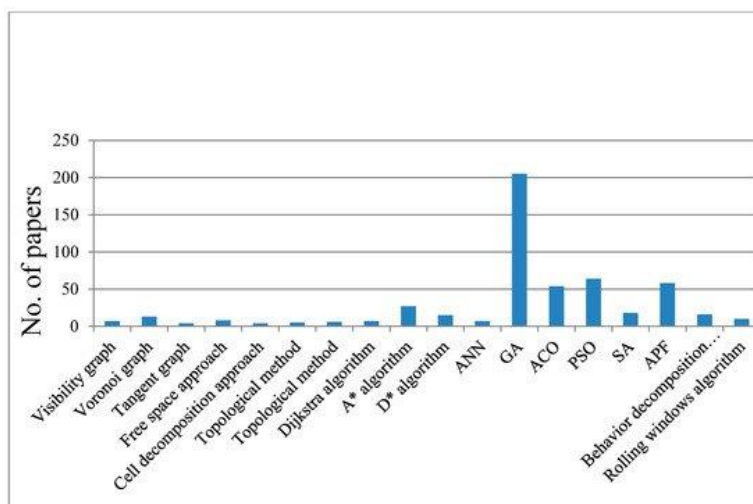


Figure 2.3: Total Number of Paper Retrieved from the Database of Engineering Village (Zhang, Lin and Chen, 2018)

Furthermore, the mobile robot path planning algorithm can be executed in the centralized or decentralized manners. In centralized manner, all the interactions between the robots are considered using a central path planner. On the other hand, under decentralized scenario, independent path planner is assigned to each of the robots and each of them will generate its own path in its own configuration space (Tang *et al.*, 2020).

There have been a lot of path planning algorithms introduced by other research which aim to solve path planning problem for MRS. In this chapter, different types of algorithms will be discussed, evaluated, and compared.

2.2 Optimization Criteria

In order to generate the optimal path for all the robots, there are a lot of aspects that must be considered in the optimization criteria when planning multi-robot paths and this often refers to multi-objective path planning problem (MRPP). Generally, the standard multi-robot multi-objective path planning optimization criteria are path length, smoothness of the path, total execution time, path safety and total energy consumption (Atiyah, Adzhar and Jaini, 2021). The path length optimization is basically referred to finding a path which has the shortest length between starting and goal locations. Secondly, the generated path can be considered smooth if the path has very low changes in the degree of direction. Thirdly, the generated path can be considered safe if the path always satisfies the safety margin with respect to the obstacles. Fourthly, the energy consumption is lesser when minimum number of robot's rotation is required for it to reach its destination.

Aggregating these objectives together, it will result in a multi-robot multi-objective path planning problem. We can determine whether the paths generated are optimal by checking whether any of the criteria had been satisfied. The idea is to obtain a feasible and optimal or near-optimal path if the path generated can satisfies two or more optimization criteria such as having the shortest path while free from any collision and so on.

2.3 Multi-robot Path Planning Algorithm

As mentioned, there are various type of MRPP algorithms that had been researched and used by many researchers and they can be divided into two

methods which are the classical method and the heuristic method. In the following subsection, the commonly used classical and heuristic approaches and algorithms will be researched, discussed, evaluated, and compared.

2.3.1 Classical Method

The classical path planning methods either find a solution or determine that no solution exists. Besides that, these methods often get stuck in local optimum which searching for path. As mentioned earlier, the disadvantages of the classical method are the high computational intensity and its inability to cope with uncertainty. This implies that the common classical methods are not suitable to be used in real life application due to its natural characteristic that is unpredictable and uncertain. Example of classical methods are sampling-based method, potential field method and probabilistic roadmap method.

2.3.1.1 Artificial Potential Field (APF) Method

The potential field algorithm also known as the virtual force field method is one of the most traditional algorithm used for path planning problem. It is a real time obstacles avoidance approach that was introduced in 1986 by Khatib (1986).

In the method of potential field, the environment space is assumed to be filled with virtual potential field. The overall idea of APF is to control the motion or movement of robots in the environment using virtual potential field as shown in Figure 2.4. The robots, target point and obstacles in the environment are given different potential. The goal position or the target point is treated as low potential points while the obstacles and the robots are treated as high potential point. The goal point in the environment which have low potential produce an attractive force to pull the robots to approach the goal point while the obstacles and other robots which have high potential produce repulsive force to prevent collision between robots and obstacles (Wu, Su and Li, 2019). Besides, the magnitude of the repulsive force is inversely proportional to the robot-to-obstacle distance. This means that whenever the robot is far away from the obstacles, the robot will not be affected by the repulsive force and only if the robot enters the influence area, it will be repelled away from the obstacles (Matoui *et al.*, 2017a). The resultant force of

the combined forces will generate a field with direction and magnitude as shown in Figure 2.4 which influence the robot to move toward the goal while avoiding collision with obstacles and other robots.

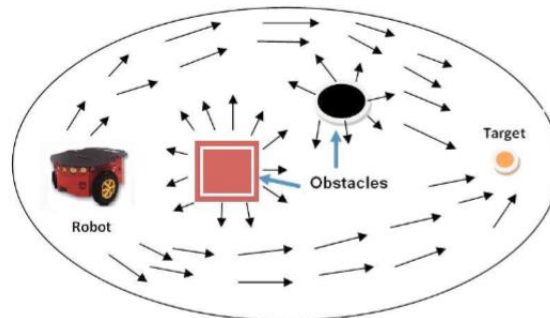


Figure 2.4: Combination of Attractive Force and Repulsive Force (Matoui *et al.*, 2017)

Due to the model's simplicity and high safety factor, multiple research projects on potential-based path planning had been done in various research. At the same time, APF has high flexibility and not only suitable to be used in static environment but also in dynamic environment where obstacles and goal location changes from time to time. APF had been used for many years and it is one of the mature approaches in solving a path planning problem for single robot. However, according to Borenstein and Koren (1989), the main drawback of APF is local minimum problem, where the robot might get stuck at the local minima before attaining the goal configuration. In year 1991, the limitations and shortcomings of APF method were further identified by Borenstein and Koren (1989). According to this Borenstein and Koren, it was found out that APF has four significant problems such as (1) the local minima trapping problem, (2) inability to deal with arbitrarily shaped obstacles, (3) the problem where no passage can be found between closely arranged obstacles, and (4) path oscillation problem with the presence of obstacles or in narrow passages. A local minimum as shown in Figure 2.5 refers to the situation where the attractive force generated by the goal position on the robot is balanced with the repulsive force generated by the obstacles on the robot. This causes the robot loses its ability to move and could not reach the goal position without any other external force (Sun *et al.*, 2019). A path

oscillation problem as shown in Figure 2.6 occurs when the angle between repulsive and attractive force is close to 180 degrees. In this case, when the robot come close to an obstacle, the repulsive force acting on the robot which is larger than the attractive force repels the robot. However, after moving away from the obstacle, the attractive force would become greater than the repulsive force and attract the robot toward the obstacle. This phenomenon will be repeated until the angle become smaller and it will only be solved once the angle between the repulsive and attractive force become smaller (Wu, Su and Li, 2019).

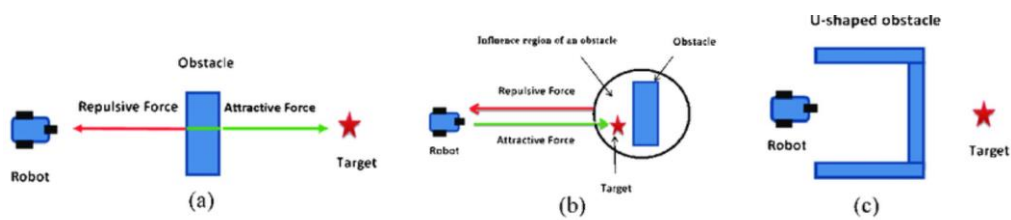


Figure 2.5: Illustration of Local Minimum Cases (Abdalla, Abed and Ahmed, 2017)

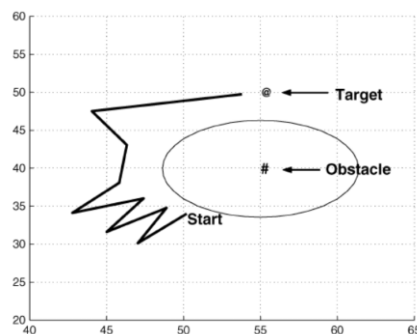


Figure 2.6: Illustration of Path Oscillation Case (Jing Ren, McIsaac and Patel, 2006)

To date, some efforts had been made in research to solve these limitations for single mobile robot. In recent research, Zhu, Yan and Xing (2006) proposed hybrid approach by implementing APF with simulated annealing (SA). In this paper, the APF is mainly used to drive the robot toward the goal position while the SA algorithm is used for the robot to escape local minima. From the result, it was shown that by applying SA together with APF, the robot could avoid local minima problem as shown in Figure 2.7. However,

the experiment was only done with simple-shaped obstacles, and the ability of the robot to escape local minimum trapping problem in complex obstacles environment such as trapping inside U-shaped obstacles was not discussed in the paper.

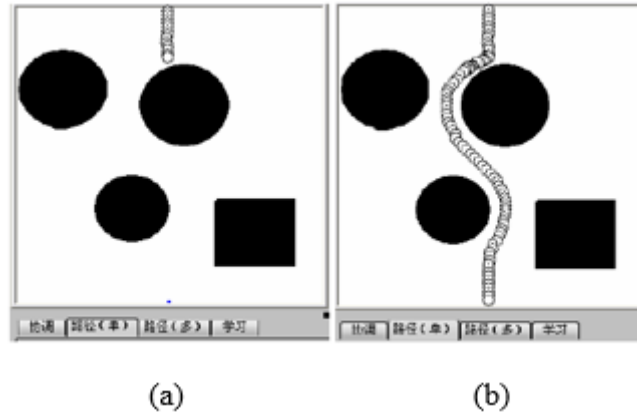


Figure 2.7: (a) Simulation with APF Which Local Minimum Occurs, (b) Simulation with APF and SA Where SA Able to Help the Robot to Escape Local Minimum. (Zhu, Yan and Xing, 2006)

In the case of implementing APF in a MRS, repulsive field was set between the robots to avoid collision between robots. Wu, Su and Li (2019) proposed an Improved APF by introducing a new gain constraint and a random factor which respectively suppressed the path oscillation problem and avoid local minimum. In Wu, Su and Li's research, four robots were implemented, and these robots were required to travel from different starting location to a same goal location without colliding with obstacles. As mentioned earlier, the path oscillation mostly likely to occur when the angle between repulsive force and attractive force is close to 180 degrees. To solve the path oscillation problem, the gain constraint was added to reduce the angle between the repulsive force and attractive force so that the oscillation problem can be suppressed. On the other hand, a random factor or an extra random force with random direction and magnitude was added into the resultant force of the robots. This extra force relieves the robot from local minimum trapping problem. Other than that, to further alleviate the path oscillation problem and optimized the path, B-spline curve optimization method was used to further smoothen the path.

In their simulation result as shown in Figure 2.8, three different results using different algorithms were simulated. The first result was obtained by implementing the original APF method, the second result was obtained by implementing the Improved APF method and the third result was obtained by implementing the Improved APF method with B-spline curve optimization. The result shows that a much smoother path can be generated by implementing B-spline curve optimization into the Improved APF algorithm. However, as reported by the authors, the oscillation could not be fully eliminated and there was still certain degree of oscillation. Besides, the research was conducted to simulate the scenario where multiple robots travel to the same location, and the ability and performance of the Improved APF for multiple robots travelling to different goal locations was not tested.

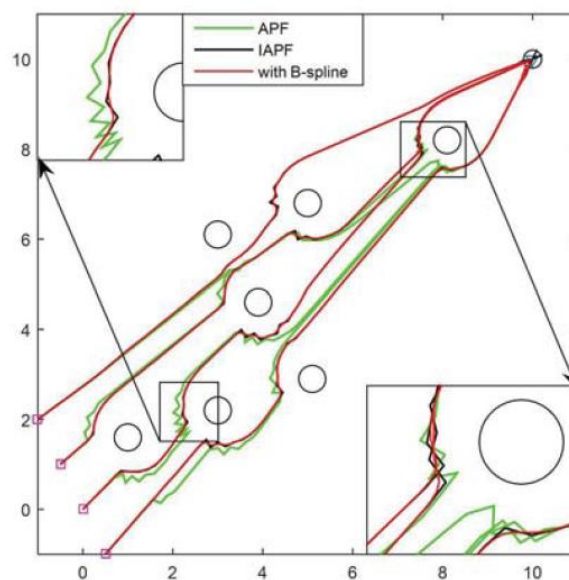


Figure 2.8: Path Planning of the three Algorithms (Wu, Su and Li, 2019)

In another research paper, Matoui et al. (2017) proposed an APF algorithm for wheeled robot with decentralized architecture. The proposed APF algorithm was used due to the possibility of implementing it in real time and dynamic environment. Decentralized architecture was used to allow cooperation between robots and to increase the flexibility and scalability of the MRS. Three Pioneer 3DX robots were used to navigate between different start location to different goal location and each of these robots plan its own trajectory according to the trajectories of other robots. By looking at their

result as shown in Figure 2.9, all three robots were able to reach its goal locations respectively without colliding with other obstacles or robots. The authors also mentioned by implementing the decentralized architecture, a greater number of robots can be added into the environment. However, based on the results, the paths generated were not the optimal path in terms of the path's length even though the robots were able to reach the goal locations without collision.

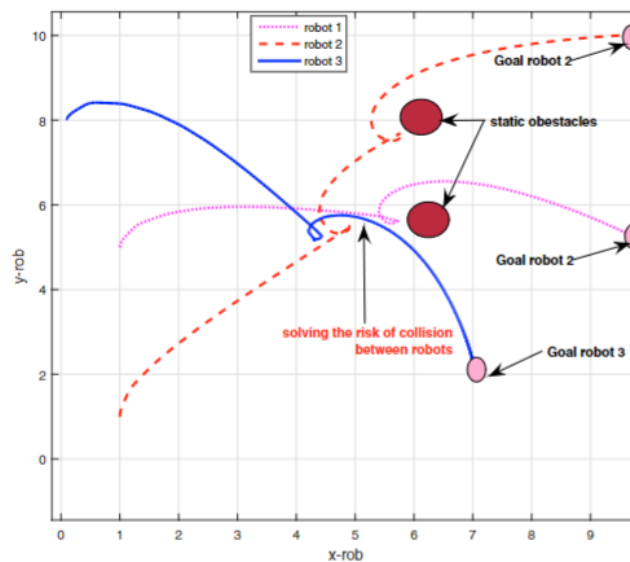


Figure 2.9: Cooperation of three Robots with The Present of Static Obstacles
(Matoui et al., 2017)

In general, APF approach is a mature path planning algorithm that was used for decades but mostly on single robot navigation, and it can be used in either static or dynamic environment. However, it has poor performance when being used in dynamic environment compared to other heuristic algorithms. Besides that, the original APF algorithm has a lot of shortcomings such as local minimum problem, path oscillation problem and the problem where no passage can be generated between two closely spaced obstacles. These limitations of the algorithm make it troublesome and complex to be solved as all limitations are difficult to be solved at once.

2.3.1.2 Rapidly Exploring Random Tree (RRT) Approach

A sampling-based approach is an approach that constructs a graph or tree by randomly sampling in the state space. It is very popular in path planning problems due to the advantages of low computational cost, better performance in complex problems, the ability to find a path in a very short time and the ability to solve high-dimensional problems (Tim Chinenov, 2019). Some path planning algorithms like the famous Dijkstra's algorithm need a pre-built graph for it to search for a path using the graph. Unlike Dijkstra's algorithm, the RRT approach could generate a graph and find a path using the graph. It generates a graph and searches for the feasible path by growing a tree that fills the entire configuration space. This tree is rooted at the starting point or starting node of the robot and a random node is generated at each iteration or each incremental expansion of the tree-growth. A simple RRT algorithm involves three steps. The first step is to randomly sample a state in state space, the second step is to select the nearest node of the random tree and the third step is to grow the tree from the nearest neighbour sampling point to a random node (Wu *et al.*, 2021). Besides that, each time a node is generated, a check must be done to ensure that the node does not lie inside an obstacle to prevent collision. This expansion of the tree stops once certain rules are met such as when the expansion of a node has reached the goal location, or the maximum iteration has been met. After that, RRT will return a feasible path as shown in Figure 2.10.

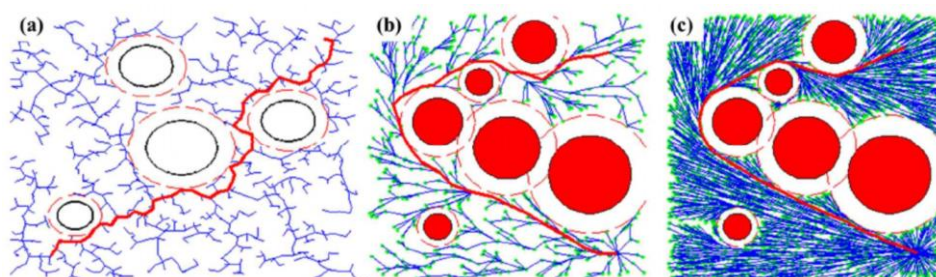


Figure 2.10: Illustration of RRT Path Planning Algorithm (Lee, Lee and Shim, 2017)

In general, RRT is an efficient path planning approach that is used to solve single robot path planning problem and various research had been done for single robot navigation problem such as the research done by Melchior and Simmons (2007) and Lee, Lee and Shim (2017). However, that is not the case in MRS. The research done on MRPP problem are still lacking today. Besides that, the simple RRT approach has some shortcomings such as low convergence speed and low search efficiency. Furthermore, the path produced by the traditional RRT algorithm is often suboptimal and lacks convergence to an optimal path because of the random generation of nodes. Regardless of its shortcoming, numerous improvements have been carried out by researchers. Several common enhancements involve effectively combining the RRT algorithm with other algorithms or implementing the optimized RRT* algorithm, which offers improved performance (Ge *et al.*, 2021).

Yang Li et al. (2013) proposed a MRPP approach using RRT* algorithm which at the same time taken into consideration of robot motion constraints and collision between robots. In contrast to the traditional RRT, RRT* able to sample a random state and generate an optimal extended state. RRT* has the working principle as the traditional RRT but with two additional processes namely “near vertices” and “rewire”. RRT* not only has the advantage of finding an initial path very quickly, it also keeps on optimizing the initial path as the number of iteration increases. The fundamental concept behind RRT* algorithm is to improve the connectivity of nodes in the trees and expedite the discovery of the goal state by employing a target-biased approach in sampling state generation. This approach aims to minimize the zigzag path behaviour commonly observed in traditional RRT, as depicted in Figure 2.11.

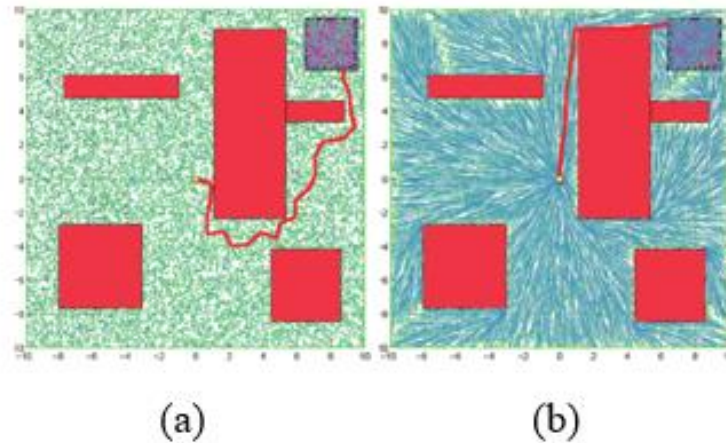


Figure 2.11: (a) Comparison of RRT and (b) RRT* Algorithm (Bohács, Gyimesi and Rózsa, 2016)

In Yang Li et al.'s research, two robots were implemented to navigate through an environment with three static obstacles from different starting point to different goal point as shown in Figure 2.13. As the robots are treated as moving obstacles, two different types of algorithms were developed to detect both static and dynamic obstacles. From the result, it shows that throughout the whole simulation, the minimum distance between the robots is larger than 3.5m which indicate that collision will not happen between the robot as the safety distance between robots was set to 1m as shown in Figure 2.12.

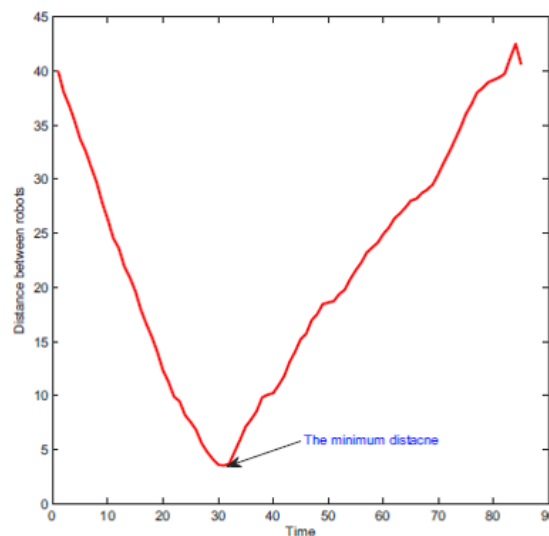


Figure 2.12: The Distance Between Robots using RRT* (Yang Li *et al.*, 2013)

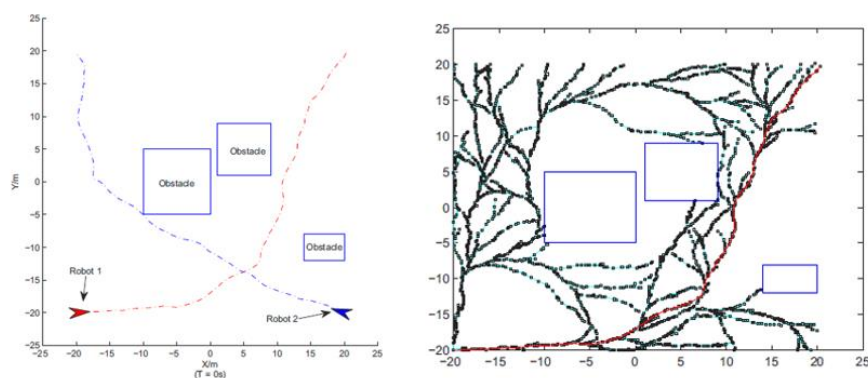


Figure 2.13: Path Generated Using RRT* for Two Mobile Robots (Yang Li *et al.*, 2013)

In another recent research presented by Connell and Manh La (2018), the ability of RRT* algorithm was tested in dynamic environment for multiple robots through planning and replanning of path. Moreover, the performance and the differences between the traditional RRT algorithm and the RRT* algorithm were further compared and evaluated by running two simulations with different algorithms in the same environment as shown in Figure 2.14. In the figure, the blue line indicates the best path found while the black line indicates the initial path found by RRT algorithm. As shown in Figure 2.14, the result with RRT* shows an obvious improvement in finding the optimal path comparing to the path generated by the traditional RRT. Besides that, it was found that the path length using RRT is 117 units while the path generated with RRT* is shorter by 14 units. This result further proved that RRT* could generate better optimal path compared to traditional RRT.

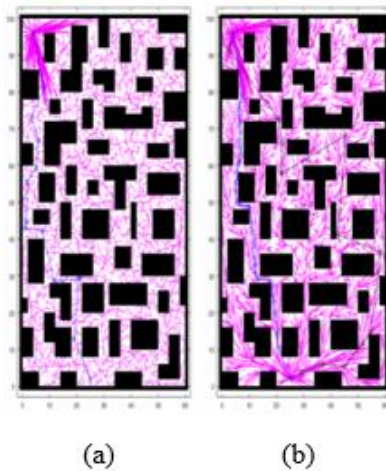


Figure 2.14: Result of RRT (a) and RRT* (b) Using 5000 Nodes (Connell and Manh La, 2018)

Besides that, to improve the ability of this algorithm working in dynamic environment, the authors also presented a dynamic replanning method for a mobile robot to avoid collision with any dynamic obstacles. This path replanning technique utilizes the rewire function, which is employed to modify the search tree by ensuring that internal vertices do not introduce unnecessary steps in any existing path when dynamic obstacles are detected. From the result, it shows that the robots were able to dynamically modify its course of action to avoid any dynamic obstacles as shown in Figure 2.15.

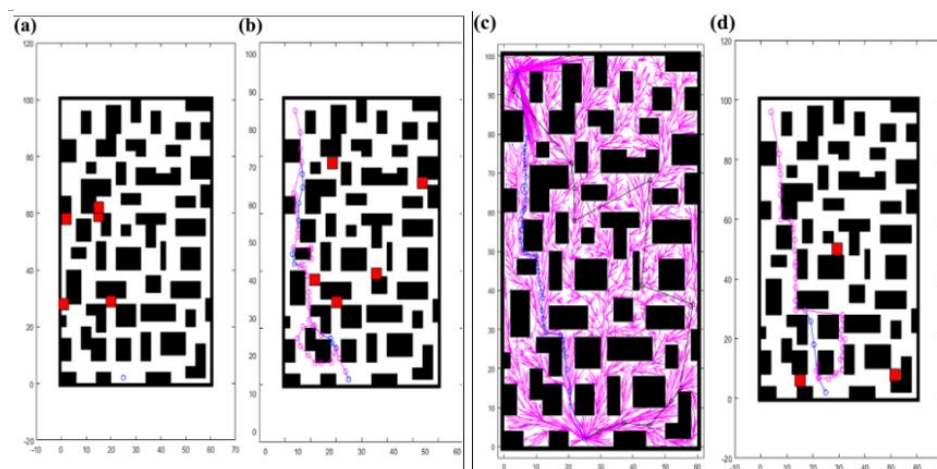


Figure 2.15: (a) Initial Position of Moving Obstacle. (b) Final Position of Moving Obstacles. (c) Result of RRT* Algorithm. (d) Final Obstacles Position and Executed Path (Connell and Manh La, 2018)

Additionally, a MRPP method was also developed by sharing nodes generated in the tree. The result shows that the proposed MRPP method with all the robots sharing the nodes can be executed with shorter amount of time compared to building multiple individual search tree for each of the robot.

Despite the fact that RRT* can quickly generate an initial path and produce an asymptotically optimal path compared to the conventional RRT algorithm, there is a trade-off in reaching this optimal path which is the execution time and the number of nodes needed. As shown in Figure 2.16, the execution time of RRT is slowly increasing as the number of nodes increases but the execution of RRT* increase exponentially as the number of nodes increases. This implies that a longer execution time and iterations are needed to obtain the optimal path.

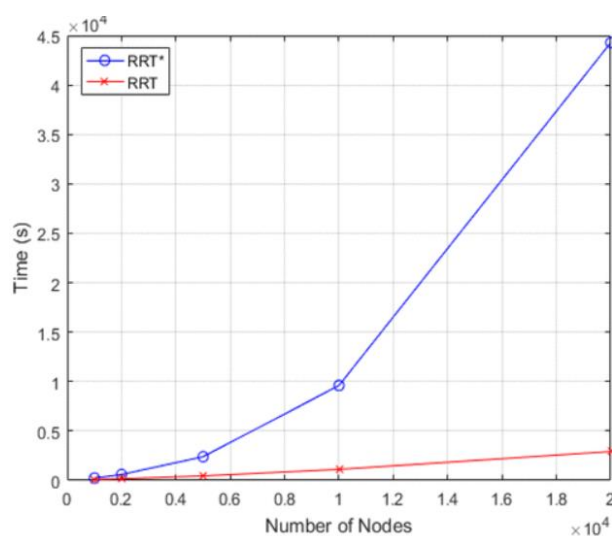


Figure 2.16: Execution time of RRT and RRT* (Connell and Manh La, 2018)

2.3.2 Heuristic Method

Due to the fact that classical path planning methods often encounter issues with local minimum trapping and exhibit high computational requirement, there are ill-suited for dynamic or complex obstacle environment. Thus, some heuristic path planning approach will be discussed in the subsequent sections. Some of the popular heuristic approaches are Genetic Algorithm (GA), Particle Swarm Optimization (PSO), Ant Colony Optimization (ACO) and so on (Kim and Khosla, 1992).

2.3.2.1 Ant Colony Optimization (ACO)

Optimization problem is one of the important problems in various fields such as industrial, engineering, and scientific areas. For instance, some of the real time examples for optimization problems are travelling salesman problems, portfolio optimization, vehicle routing and timetable scheduling. To date, a lot of optimization algorithms had been developed to solve these optimization problems and Ant Colony Optimization is one of the examples of optimization algorithm (Awan-Ur-Rahman, 2020). Ant Colony Optimization (ACO) is a swarm intelligent algorithm that implement swarm behaviour and it is a probabilistic technique in finding optimal path. Initially, ACO was used in solving the travelling salesman problem. Today, ACO is also used to handle navigation problem for mobile robots to generate an effective path with obstacle avoidance.

ACO is inspired by an analogy on behaviour of an ant colony when sourcing food as shown in Figure 2.17. Essentially, the communication between the ants enables them to locate the shortest route between the food source to their nest. To obtain the optimal path with the shortest distance, the communications of ants are done by the means of pheromone trails (Patle *et al.*, 2019). As the ant travel, some amount of pheromone is dropped on the path. As the pheromone evaporates, the shortest path that the ant travel will leave a higher density of pheromone deposited on the ground. At the same time, the more ants travel using a certain trail, the more attractive this trail become, causing even more ant to follow it due to high density of pheromone. Eventually, the optimal path can be obtained by evaluating the amount of pheromone deposited on a certain path (Mac *et al.*, 2016). However, according to Mac *et al.* (2016), the traditional ACO has disadvantage of long time to reach optimal solution when used in large size problem.

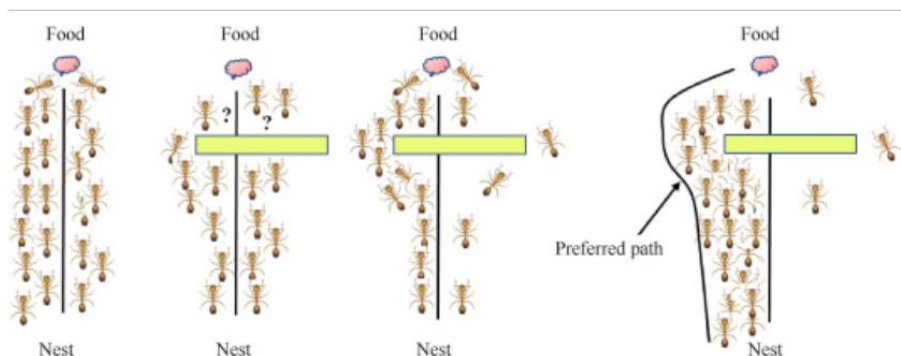


Figure 2.17: Illustration of ACO (Patle *et al.*, 2019)

TAN (2007) proposed an optimal path planning based on ACO and Dijkstra algorithm for real time application. The Dijkstra algorithm was employed to identify a sub-optimal path while considering collision avoidance, and subsequently, the ACO algorithm was applied to optimize this sub-optimal path, resulting in the generation of a globally optimal path. The result in Figure 2.18 shows that ACO indeed effective to be implemented in real time application. Additionally, the performances of ACO and GA were compared, it was verified that ACO has better performance compared to GA in terms of convergence speed, dynamic convergence behaviour and computational efficiency as shown in Table 2.1. For instance, ACO algorithm only took average of 0.00059 seconds for one iteration and the total number of iterations needed was only 175 iterations. On the contrary, GA algorithm took 0.00067 seconds per iteration and the number of iterations required was a lot more than ACO around 912 iterations.

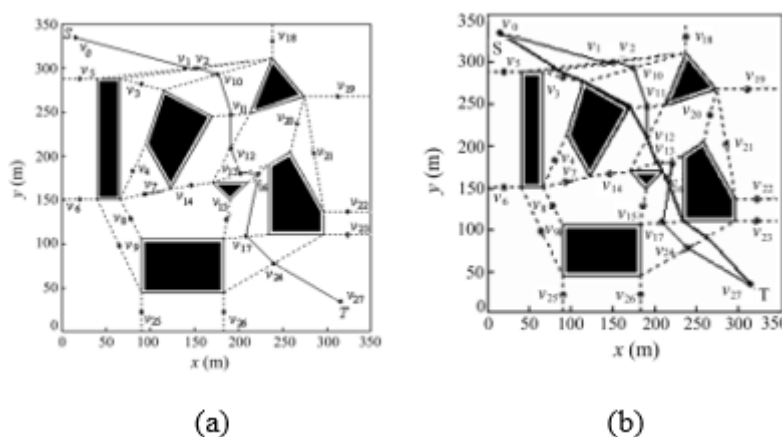


Figure 2.18: (a) Sub-optimal Path Using Dijkstra Algorithm, (b) Globally Optimal Path with ACO (TAN, 2007)

Table 2.1: Comparison of Computation Efficiency of ACO and GA (TAN, 2007)

Algorithm	Average CPU time per iteration (sec.)	Average number of iterations needed for convergence	Average CPU time needed for obtaining optimal solution (sec.)
ACS algorithm	0.00059	175	0.1033
Real-coded GA	0.00067	912	0.6110

In Liu, Mao and Yu (2006), a path planning algorithm based on ACO was proposed to solve MRPP problem. In Liu, Mao, and Yu's paper, a collision avoidance strategy was adopted for various robots to avoid collision with each other in a static environment. In ACO algorithm, deadlock is one of the problems that occur in the traditional ACO. Deadlock refers to the scenario where the robot is stuck and loses its moveable possibility. Deadlock problem as shown in Figure 2.19 normally occur when an ant enters a location that is surrounded by obstacles when searching for a path. This scenario causes the ant to lose its capability to go forward and this is known as the deadlock problem. As such, a special penalty function was introduced by Liu, Mao, and Yu to solve the deadlock problem. Whenever an ant finds a dead corner, a penalty function was used to decrease the phenomenon intensity of the edges around the dead corner. Thus, the other ants will ignore those edges in the coming iteration and deadlock could be avoided.

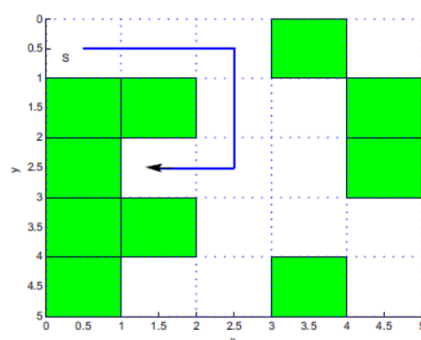


Figure 2.19: Illustration of Dead Corner and Route Deadlock (Liu, Mao and Yu, 2006)

Besides that, the algorithm was executed in distributed manner where each of the robot plan its own path from its start location to its goal. The performance and the planned trajectory of two mobile robots using two different approaches which are the ACO, and GA were compared, and it was

found that a more reasonable path can be generated using the proposed ACO algorithm. The experiment was conducted in two different environments where more obstacle was added in one of them. From the result shown in Figure 2.20 and Table 2.2, the path generated using ACO is observed to be much shorter than the one generated by GA.

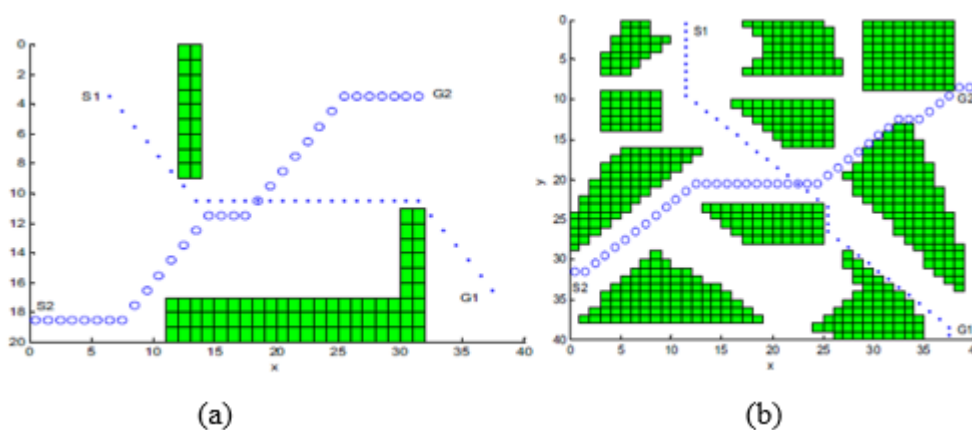


Figure 2.20: (a) Environment I and (b) Environment II (Liu, Mao and Yu, 2006)

Table 2.2: Comparison of Path Length Generated Using ACO and GA (Liu, Mao and Yu, 2006)

Approach	Environment I		Environment II	
	$L_1(S_1, G_1)$	$L_2(S_2, G_2)$	$L_1(S_1, G_1)$	$L_2(S_2, G_2)$
Improved ant colony algorithm	36m	37m	50m	49m
Genetic algorithm in reference [6]	61m	41m	53m	57m

2.3.2.2 Particle Swarm Optimization (PSO)

Like ACO, Particle Swarm Optimization is also one of the biologically inspired algorithms. It is a meta-heuristic optimization algorithm which is inspired by the natural behaviour of creatures such as flock of birds or school of fishes, developed in 1995 by Eberhard and Kennedy (Eberhart and Kennedy, no date). PSO mimic the social behaviour of animals but unlike the real natural behaviour of animals which normally require a leader within a group to do certain task or to reach certain target, PSO does not require any leader within

the group. For instance, when a flock of birds go to source for food, they do not go with a certain leader but tend to go with the members which are nearest to the food location. PSO uses the concept of social and cognitive behaviour of animal group where sharing of information is done among the group of animals to increase the survival advantages. Taking an example of a bird searching for food, the chance of finding food will increase if the bird works with the flock where there is mutual sharing of the best information among the flock which then help the flock to locate the best place to hunt. Similarly, the concept of social interaction of animals in PSO is implemented to search for optimal path for multi-robot navigation and it is gaining more popularity in solving complex problem such as multi-robot navigation problem in real time application due to its advantages of fast searching speed, few parameters to adjust, easy implementation, and simple structure. However, when used in path planning problem, PSO has a few disadvantages such as low convergence precision, reduced particle diversity and local optima trapping issue (Li *et al.*, 2020).

In PSO, the problem's working space is initially populated with random particles, where a swarm of particles is assigned random positions and velocities. These particles then explore the space iteratively, updating their position in search of the optimal solution. Particles will move in the search space based on their respective velocity and this velocity is dependent on several variables such as inertial, its previous velocity, personal best solution, and global best solution as shown in Figure 2.21. In every iteration, each of the particles are updated with two best values. The first best value indicates the best solution obtained by the particle until the current iteration which refer to the personal best (pBest) On the other hand, the second-best value is the best value obtained by any particles until the current iteration and it is referred to global best (gBest). These two best values affect the next position for each of the particles. The velocity or movement of each of the particles are affected by some factors such as random factors, inertial constant and constriction factor. These factors are responsible on the exploration and exploitation of the swarm (Das and Jena, 2020).

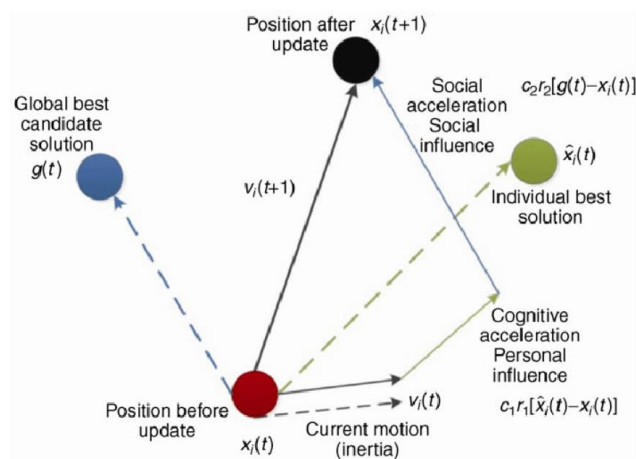


Figure 2.21: Illustration of PSO (Iran Macedo, 2018)

Maryam Yarmohamadi and Hossein Erfani (2011) presented a PSO path planning algorithm as a feasible approach for self-organized control of a single mobile robot to avoid colliding with obstacles during the trajectory. In their work, the initial position (x_r, y_r) of that one robot is initialized in a random coordinate at time (t) and the next position of the robot (x_r', y_r') in the next iteration time $(t+1)$ is calculated using PSO. The next position of the robot will be calculated in every iteration until the robot reach its goal. In their work, the implementation of PSO involves the initial population of random particles being generated around the current position of the robot. At the end of the algorithm for that iteration, the particle with the best position or $gBest$ will be chosen to be the next position of the robot. The objective function of their work is to calculate the shortest distance between the current location to the goal location using Euclidean distance. Besides that, an objective function will be developed by adding a penalty function to avoid the obstacle. If the robot seen an obstacle in front of it during the trajectory, the robot will decide the direction and rotate either left or right. During the experiment, a single robot was employed to navigate an environment containing static and dynamic obstacles, as depicted in Figure 2.22. The starting location was indicated by a blue dot, the goal location by a red dot, dynamic obstacles by green circles, static obstacles by white circles, and the generated path was represented by a line. From the simulation result, it was found that the robot can successfully navigate from start location to the goal location without any collision and it

prove that PSO can be effectively implemented in complex and dynamic environment.

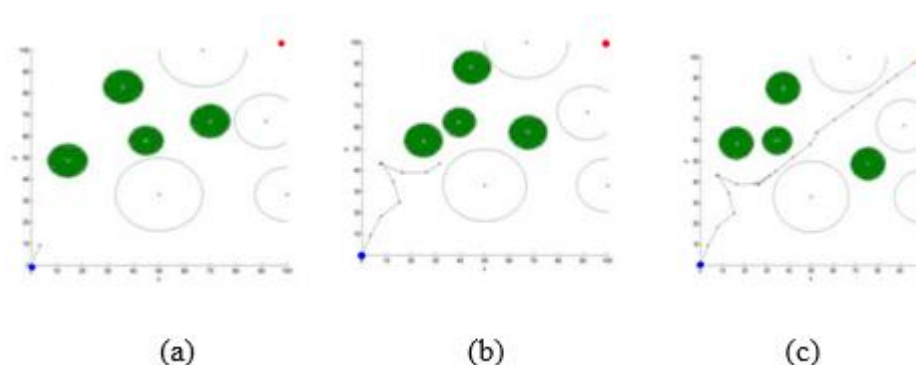


Figure 2.22: (a) In Start Time, (b) In Middle Time, (c) In End Time (Maryam Yarmohamadi and Hossein Erfani, 2011)

As the number of robots used in a MRS increase, the control of the MRS become infeasible and unreliable. Swarm behaviour-based approach has been used to effectively solve this problem thanks to its properties of robustness, scalability, and flexibility. Ayari and Bouamama (2017) proposed a distributed or decentralised MRPP algorithm using Dynamic Distributed Particle Swarm Optimization (D²PSO). In their work, the authors address the main drawback of traditional PSO which is the local optimal problem and to solve this problem, they introduced two new parameters into the classic PSO which are the Local Optima Detector for gbest and Local Optima Detector for pbest. These two parameters are employed to track the number of consecutive iterations in which there is no improvement in pbest and gbest position. This indicates that particles that fail to improve their pbest will no longer contribute to gbest. Thus, it also means that the particles are saturated and need extra thrust. When a gBest is not improving for a predefined successive iteration, it might be due to local optima trapping problem. D²PSO proposed in their paper is to give that extra thrust to push the trapped particles in local optimal out from the local optima position by heading particles toward a better and unexplored region. In their experiment, ten robots were used in a known environment and the D²PSO algorithm was run with 200 iterations as shown in Figure 2.23. Additionally, the performance of the D²PSO was compared to the Distributed PSO algorithm. The result in Table 2.3 shows that the D²PSO

perform better in escaping local optima and it can be used for large number of robots. Besides, it also shows that D^2PSO can generate shorter path than Distributed PSO using population size, M of 300 in general.

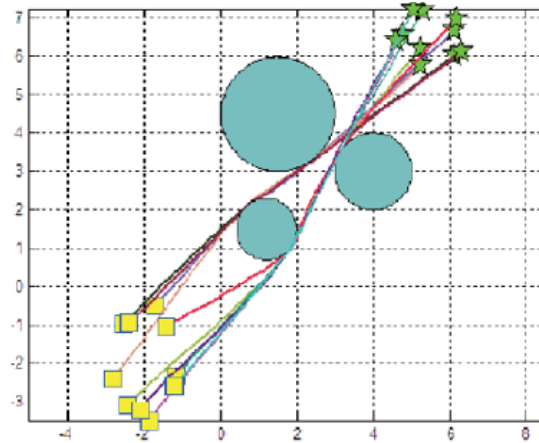


Figure 2.23: Result Obtained using D^2PSO (Ayari and Bouamama, 2017)

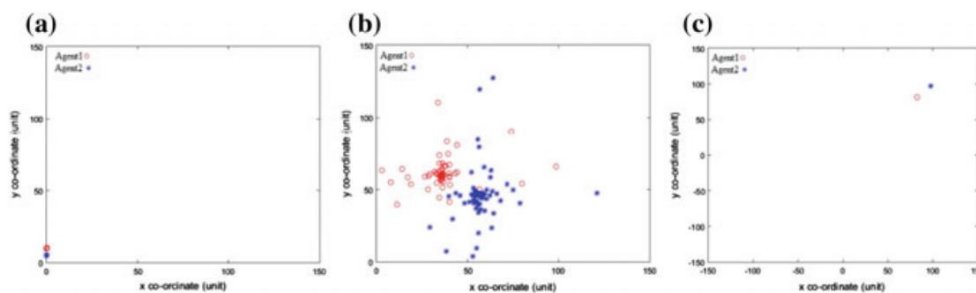
Table 2.3: Path Length Comparison of D^2PSO and $DPSO$ (Ayari and Bouamama, 2017)

10 robots $M=300$		
	<i>Distributed PSO</i>	<i>D^2PSO</i>
Robot1	12.2421	12.1664
Robot2	11.0449	10.6679
Robot3	12.4518	12.134
Robot4	11.1951	7.8591
Robot5	11.6412	8.1257
Robot6	12.2114	9.1098
Robot7	9.4232	7.6011
Robot8	10.9119	8.6738
Robot9	11.1931	8.4381
Robot10	12.3391	10.6956

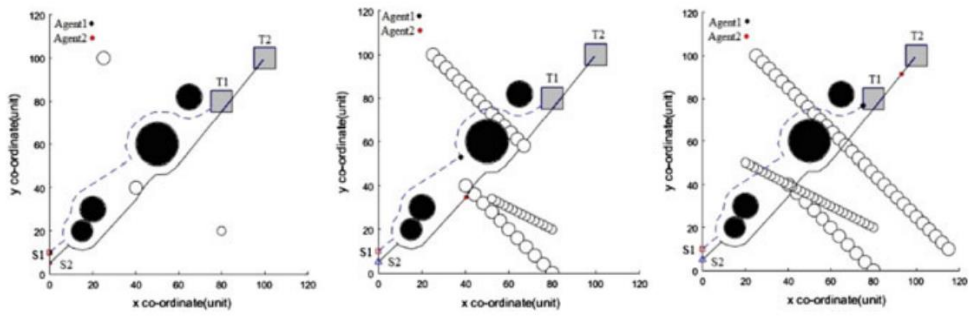
Biswas, Anavatti and Garratt (2017) proposed an obstacles avoidance method for MRPP using Simultaneous Replanning Vectorized Particle Swarm Optimization (SRVPSO) that focus on effective and deficient obstacle avoidance. The concept of simultaneous replanning was implemented into the MRPP in order to avoid collision. In their algorithm, the robots used in the environment considered each other as dynamic obstacles and collision avoidance strategy was then implemented. The collision avoidance method

they used are divided into two steps. If the particles initialized by PSO algorithm are within a collision zone, a very high cost will be assigned to the fitness function or the objective function of the particles and this is the first step. As a result, the velocity of those particles that are within the collision zone will be rapidly increased which make the particles to leave the collision zone and increment in the velocity is the second step. In addition, their research utilizes Vectorized PSO, which differs from the classical PSO in terms of evaluating the objective function. In vectorized PSO, the objective function evaluates all points in the search space simultaneously, whereas in the classical PSO, the objective function evaluates one point at a time. The vectorized PSO is used instead of the classic PSO to reduce the programming code as it does not require the code to be run in loops which make it run faster than the classic PSO which require loop. Other than that, the distance between the particles and the obstacles which denoted as D_{obs}^a were calculated and evaluated at each iteration to maintain a collision free path and this distance must be bigger than 0 to avoid collision. Whenever this distance is calculated to be negative, a high cost will be assigned into the objective function of the particles and these particles will not be considered as the best fitness function to avoid collision. In simulation as shown in Figure 2.24 and Figure 2.25, two robots were used in an environment filled with static and dynamic obstacles. The results demonstrate that the SRVPSO algorithm effectively generates paths with a high level of safety by avoiding obstacles, while also optimizing the paths efficiently.

$$D_{obs}^a \leq 0$$

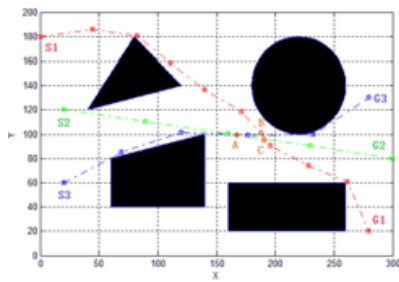


Figure

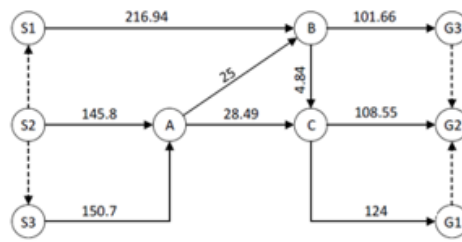


Figure

Han,



(a)



(b)

Figure

Table

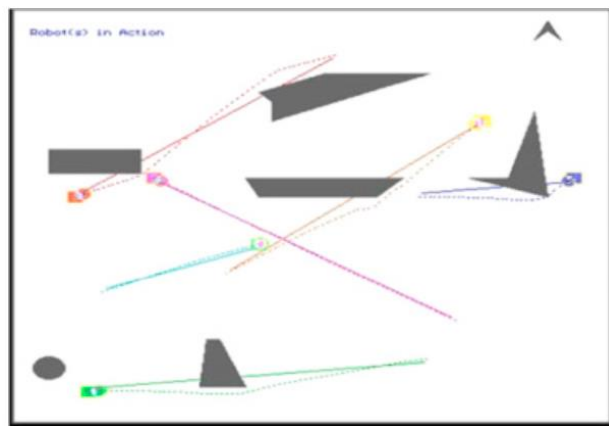
Node(i)	ET(i)	LT(i)
<i>S1</i>	0	0
<i>B</i>	216.94	216.94
<i>C</i>	221.78	221.78
<i>G1</i>	345.78	345.78
<i>S2</i>	0	0
<i>A</i>	150.7	191.94
<i>G2</i>	345.78	345.78
<i>S3</i>	0	41.24
<i>G3</i>	318.6	345.78

Table

Robot Number	Node 1	Pass Time	Node 2	Pass Time	Node 3	Pass Time
1	<i>B</i>	216.94	<i>C</i>	221.78	<i>G1</i>	345.78
2	<i>A</i>	before 145.7	<i>C</i>	before 216.78	<i>G2</i>	before 345.78
3	<i>A</i>	before 186.94	<i>B</i>	before 211.94	<i>G3</i>	before 345.79

Different

In



Figure

Table

Robot number	No of step required to goal in IPSO-IGSA	No of step required to goal in IPSO	No of step required to goal in IGSA
1	21	24	23
2	21	25	31
3	21	23	27
4	11	14	13
5	21	25	25
6	11	13	15

Table

Robot number	IPSO-IGSA		IPSO		IGSA	
	Avg. trajectory		Avg. trajectory		Avg. trajectory	
	distance travelled	Time (s)	distance travelled	Time (s)	distance travelled	Time (s)
1	329.665	15.234	359.267	17.432	387.873	24.347
2	308.665	26.782	321.456	29.637	336.479	42.173
3	291.006	45.362	308.665	48.538	321.483	52.467
4	250.787	51.263	289.462	56.378	291.697	66.879
5	293.626	58.234	267.556	62.469	272.463	78.467
6	232.79	61.221	245.662	69.384	248.382	91.832

2.4 Summary

In

Table

Classical	Rapidly	<ul style="list-style-type: none"> - Able to generate the optimum path or the shortest path with increase of iterations. - The number of iterations required, and execution time increase exponentially in solving MRPP problem. - Increase computational cost with increased number of robots. - Lack of research done for MRPP problem.
	Artificial	<ul style="list-style-type: none"> - Good performance in obstacle avoidance in both static and dynamic environment. - Local optimum trapping problem. - Path oscillation problem. - Weak in finding path between closely arranged obstacles.

		<ul style="list-style-type: none"> - Lack of research done for MRPP problem.
Heuristic	Ant	<ul style="list-style-type: none"> - Effective in finding global optima. - Suitable to be implemented in dynamic environment. - Slow convergence. - High memory usage.
	Particle	<ul style="list-style-type: none"> - Fast convergence. - Simple implementation as there is only a few parameters need to be adjusted. - Prone to trapping in local optima

Based

CHAPTER 3

METHODOLOGY AND WORK PLAN

3.1 Introduction

This

3.2 Proposed Multi-robot Path Planning Algorithm

Based

3.2.1 Assumptions

The

- Starting position, current position, and destination for all the robots are known in each reference coordinate system.
- Position of static obstacles are known in each reference coordinate system.
- Each robot computes its complete path from starting position to its destination before the navigation.
- All robot used is assumed to be differential drive robot and each of them are identical to each other.

3.2.2 Classical PSO Algorithm (PSO)

As mentioned in Chapter 2, PSO is an optimization algorithm based on social behaviour of group of animals. It performs path planning by first initializing a population of random particles which distributed uniformly around the robot search space and these particles are assigned with random velocity initially and they constantly change its velocity and position dynamically during the optimization process until they reach the goal position. At every iteration, the particles change their position by updating their velocity and this velocity are influence by several factors such as the particles' experience, particle current motion, influence of the whole swarm as shown in Figure 2.21. Each particles update its position and velocity in every iteration based on equations $V_i(t +$

$$V_i(t+1) = V_i(t) + c_1\varphi_1(x_{PBest_i} - x_i(t)) + c_2\varphi_2(x_{GBest} - x_i(t)) \quad (3.1)$$

and $x_i(t+1) = x_i(t) + V_i(t+1)$ (3.2). The velocity component in the equation provides a track on previous direction that the particles travelled. The cognitive element then compares the current particles' performance with the previous best performance. Lastly, the social element compares the current particles' performance with the previous global best performance of the swarm.

Particle's Velocity Equation:

$$V_i(t+1) = \underbrace{V_i(t)}_{\text{Velocity}} + \underbrace{c_1\varphi_1(x_{PBest_i} - x_i(t))}_{\text{Cognitive Element}} + \underbrace{c_2\varphi_2(x_{GBest} - x_i(t))}_{\text{Social Element}} \quad (3.1)$$

where

- $V_i(t+1)$ = Velocity of i^{th} particle for next iteration, $t+1$
- $V_i(t)$ = Current velocity of i^{th} particle at current iteration, t
- $x_i(t)$ = Current position of i^{th} particle at current iteration, t
- x_{PBest_i} = $(x_i(t), y_i(t))$ is the current personal best position of i^{th} particle
- x_{GBest} = $(x_{GBest}(t), y_{GBest}(t))$ is the current global best position achieved by the swarm.
- c_1 = Cognitive parameter
- c_2 = Social parameter
- φ_1, φ_2 = Independent variables uniformly distributed in $[0,1]$

Particle's Position Equation:

$$x_i(t+1) = x_i(t) + V_i(t+1) \quad (3.2)$$

where

- $x_i(t+1)$ = Position of i^{th} particle for next iteration, $t+1$
- $x_i(t)$ = Current position of i^{th} particle at current iteration, t
- $V_i(t+1)$ = Velocity of i^{th} particle for next iteration, $t+1$

3.2.3 Modified PSO Algorithm (MPSO)

Until today, different version of the PSO algorithm have been suggested by researchers to improve its effectiveness, as discussed in Chapter 2. According to research done by Das, Behera and Panigrahi (2016), balancing the exploration and exploitation of the algorithm is crucial for enhancing its performance. This can be achieved by including an inertial weight factor, designated as ω , in the original particle's velocity equation as shown in Equation $V_i(t+1) = \omega V_i(t) + c_1 \varphi_1(x_{PBest_i} - x_i(t)) + c_2 \varphi_2(x_{GBest} - x_i(t))$ (3.3). The value of the ω coefficient plays a significant role in balancing the exploration and exploitation of the particles, and it greatly influences the convergence behavior of the PSO algorithm. Adjusting the ω value dynamically can modify the searching capability of the algorithm.

$$V_i(t+1) = \underbrace{\omega V_i(t)}_{\text{Inertial}} + \underbrace{c_1 \varphi_1(x_{PBest_i} - x_i(t))}_{\text{Cognitive Element}} + \underbrace{c_2 \varphi_2(x_{GBest} - x_i(t))}_{\text{Social Element}} \quad (3.3)$$

Where

ω = Inertial weight factor

According to equation $V_i(t+1) = \omega V_i(t) + c_1 \varphi_1(x_{PBest_i} - x_i(t)) + c_2 \varphi_2(x_{GBest} - x_i(t))$ (3.3), there are three problem dependent parameters which need to be tuned to get the desired result. These parameters are the inertial weight factor (ω), and the acceleration coefficient tied to the cognitive (c_1) and the social (c_2) parameters. ω is important in balancing the global exploration and exploitation or local exploration ability of the particles. ω greatly influence the convergence behaviour of the PSO algorithm. It is important as it serves to balance between the exploration and exploitation or the local exploration ability of the particles. A well-balanced local exploration ability enables the robot to find the optimal path accurately and rapidly. According to researches done by Shi and Eberhart (n.d.), Das, Behera and Panigrahi (2016), a large value of ω able to provides a wide or global search while a small value of ω facilitates fine or local search. It was found that the

searching ability of the algorithm can be dynamically adjusted by changing the value of ω . It was proved that by gradually decreasing ω throughout the optimization process would improve the particles' ability between local and global exploration. Based on the aforementioned research paper, the best value for maximum and minimum ω is 0.95 and 0.4 respectively. Therefore, in this project, a linearly decreasing ω technique will be adopted, varying from start to end of the algorithm at 0.4 and 0.95, respectively. The value for the ω can be determined using the equation $\omega = \omega_{Start} - \frac{\omega_{Start} - \omega_{End}}{K} k$ (3.4).

$$\omega = \omega_{Start} - \frac{\omega_{Start} - \omega_{End}}{K} k \quad (3.4)$$

where

ω_{Start}	= Start value of inertial weight, (e.g 0.95)
ω_{End}	= End value of inertial weight, (e.g 0.4)
K	= Maximum number of iterations
k	= Current iteration

Besides that, the acceleration coefficient c_1 and c_2 also have great effect on the performance of the algorithm. A higher c_1 attracts the particle toward its personal best position while a higher c_2 attracts the particle toward the global best position. Some researchers adjust these parameters by trial-and-error technique until a desired output is achieved. However, based on Das, Behera and Panigrahi (2016), a large value of c_2 compared to c_1 will lead the particle to local optimum prematurely and a high c_1 compare with c_2 will result in the particles wandering within the search space. They also proved that the quality of the solution can be further improved by gradually decreasing the cognitive element and gradually increasing the social element as the iteration increase. This tuning can be done using equations $c_1 = c_{1i} - \left(\frac{c_{1i} - c_{1f}}{K}\right) k$

$$(3.5) \text{ and } c_2 = c_{2i} + \left(\frac{c_{2i} - c_{2f}}{K}\right) k \quad (3.6).$$

$$c_1 = c_{1i} - \left(\frac{c_{1i} - c_{1f}}{K}\right) k \quad (3.5)$$

$$c_2 = c_{2i} + \left(\frac{c_{2i} - c_{2f}}{K} \right) k \quad (3.6)$$

where

c_{1i}	= Initial value for cognitive component
c_{1f}	= Final value for cognitive component
c_{2i}	= Initial value for social component
c_{2f}	= Final value for social component

3.2.3.1 New Path Planning Scheme Implemented in MPSO

In most of the existing path planning problem that utilize the PSO algorithm, the particles swarm is first initialized around the robot starting location, and the particles' velocity and position are updated in each iteration. In each of the iteration, one robot's waypoint will be determined by the algorithm based on the global best position found by the particle swarm. This process is repeated until all the particles have converged to the robot's target position. This means that if a robot's trajectory requires ten waypoints from its starting position to its target position, the PSO algorithm is executed for one execution of 10 iterations, and it will be terminated after 10 iterations are completed. In this report, a new PSO path planning scheme inspired by the works of the authors in Das et al. (2016) is introduced into the algorithm. Instead of determining one waypoint of the robots in each iteration of the PSO algorithm, which is done in most of the research papers, this project proposes a new path planning scheme where the algorithm is run once with a few iterations to determine one waypoint of the robots. For a robot's complete path from starting location to its destination, consisting of five waypoints, the PSO algorithm will be executed five times. In contrast to the normal PSO approach, the proposed PSO employs a different strategy where the particles' swarm is not initialized at the robot starting location, but rather initialized within a predefined search space. This algorithm with the new path planning scheme is called the modified PSO algorithm (MPSO).

3.2.3.2 Objective Functions

Creating an appropriate fitness function is essential in finding the best path for each robot while fulfilling environmental requirements like minimizing energy consumption, execution time, and overall path length. In response to this, two main objective functions are considered to tackle the path planning problem. The first objective function aims to generate the shortest possible path between the robot initial position and the goal position while the second objective function ensures that there is no collision between robot and static obstacles.

The determination of the successive waypoints for the robots is contingent on the particles' ability to search and convergence to a global optimum position, while the global optimum position is the position found by the particles swarm within the local search space which has the minimum distance to the target position. This can be accomplished by utilizing the first fitness function, F_1 , expressed in equation $F_1 = \sum_{j=1}^{nr} \sqrt{(x_j^{\text{next}} - x_j^{\text{goal}})^2 + (y_j^{\text{next}} - y_j^{\text{goal}})^2}$ (3.7), which determine the successive waypoint with the minimum distance to the destination of the robot by calculating the Euclidean distance between each of the particle's location and the goal location. The particle with the minimum distance will be considered as the next waypoint for the robot.

$$F_1 = \sum_{j=1}^{nr} \sqrt{(x_j^{\text{next}} - x_j^{\text{goal}})^2 + (y_j^{\text{next}} - y_j^{\text{goal}})^2} \quad (3.7)$$

nr	= number of robots
x_j^{next}	= x-coordinate of successive waypoint for j^{th} robot (x-coordinate of particle with minimum Euclidean distance)
y_j^{next}	= y-coordinate of successive waypoint for j^{th} robot (y-coordinate of particle with minimum Euclidean distance)
x_j^{goal}	= x-coordinate of target position for j^{th} robot
y_j^{goal}	= y-coordinate of target position for j^{th} robot

In terms of collision avoidance, the distance between the particles and the obstacles must not be too close to each other when determining the waypoint for the robots. In each iteration of the algorithm, the particles aim to converge toward a global optimum position within the local search space while avoiding any overlapping with obstacles. Each particle that overlaps with any obstacles are reinitialized to a new position until it is no longer overlapping with any obstacles. To avoid any particle getting too close to any of the obstacle, the second fitness function, F_2 , can be implemented as shown

$$\text{in equation } F_2 = \sum_{j=1}^{nr} \left\{ \frac{1}{\sqrt{(x_j^{\text{next}} - x_j^{\text{obs}})^2 + (y_j^{\text{next}} - y_j^{\text{obs}})^2}} \right\} \quad (3.8). \quad \text{The fitness}$$

function for obstacle avoidance can be expressed as a Euclidean distance function of the distance between the particles and static obstacles. The closer the particle is to the obstacles, the larger the fitness value. Thus, the position of the particle near any obstacle will not be considered as the global optimum point.

In addition, it should be noted that any particle with a location where the path is blocked by an obstacle as illustrated in Figure 3.1, will be deemed as infeasible point. When this happen, the fitness function will be assigned a value of infinity and the position of that particle will not be considered as the global best position.

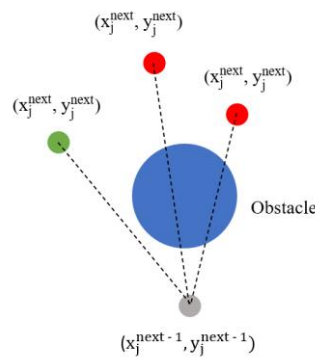


Figure 3.1: Infeasible Path When Obstacle Blocks Path

$$F_2 = \sum_{j=1}^{nr} \left\{ \frac{1}{\sqrt{(x_j^{\text{next}} - x_j^{\text{obs}})^2 + (y_j^{\text{next}} - y_j^{\text{obs}})^2}} \right\} \quad (3.8)$$

The overall objective function is obtained by summing the two objective functions as shown in equation $F = \lambda_1 F_1 + \lambda_2 F_2$ (3.9), where the λ_1 and λ_2 are the weights factors of both the fitness functions. The λ_1 and λ_2 values are adjusted in the simulation to get obtain the desired result by trial and error. Higher λ_1 indicate that F_1 is more important than and vice versa. Thus, the optimal path can be achieved by reducing the total fitness value as represented in equation $F = \lambda_1 F_1 + \lambda_2 F_2$ (3.9).

$$F = \lambda_1 F_1 + \lambda_2 F_2 \quad (3.9)$$

3.2.4 Dynamic Obstacle Avoidance

The proposed MPSO algorithm utilized in this project deviates from the normal PSO path planning approaches that employ local path planning methods, which determine paths incrementally as the robot progresses towards its goal. Instead, the proposed MPSO algorithm adopts a global path planning approach, wherein it plans complete paths for all robots prior to their navigation within a pre-existing simulated environment. During navigation, an sensor-based obstacle avoidance algorithm is implemented as the local path planner to avoid collision with dynamic obstacle. This algorithm is known as the Obstacle Avoidance Algorithm in this project. The algorithm is based on sensor data generated from sensor such as LIDAR sensor to avoid dynamic obstacles within the robot's sensing range. The algorithm uses multiple sensors attached to each robot as shown in Figure 3.2 to detect obstacles in the environment. Upon detection of dynamic obstacle, the robot adjusts its path to avoid colliding with the obstacle. The robots also treat each other as dynamic obstacles, allowing them to detect and avoid possible collisions with each other. In the simulation, three sensors are affixed to each robot, with a combined sensor angle coverage of 180 degrees. Each individual sensor covers 60 degrees of the robot's surroundings. When a sensor detects a dynamic obstacle during navigation, the sensor will be assigned with a logic value of 1 indicating that the sensor detects obstacle, and the robot adjusts its path to avoid the obstacle and will not travel in the direction of the obstructed sensor. If the sensors did not detect obstacle, the robot travels toward the direction of

the sensor that provide minimum distance between the robot's current position to the next waypoints.

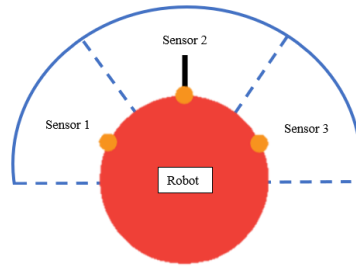


Figure 3.2: Sensor Angle Coverage of Robot

3.2.5 Flowchart of MPSO Path Planning Algorithm

Figure 3.3 illustrates the flow chart of PSO path planning algorithm for a single robot. This flowchart will be implemented on all the robots in the MRS. The pseudocode for MPSO algorithm is provided in the Appendix PseudocodeA-1.

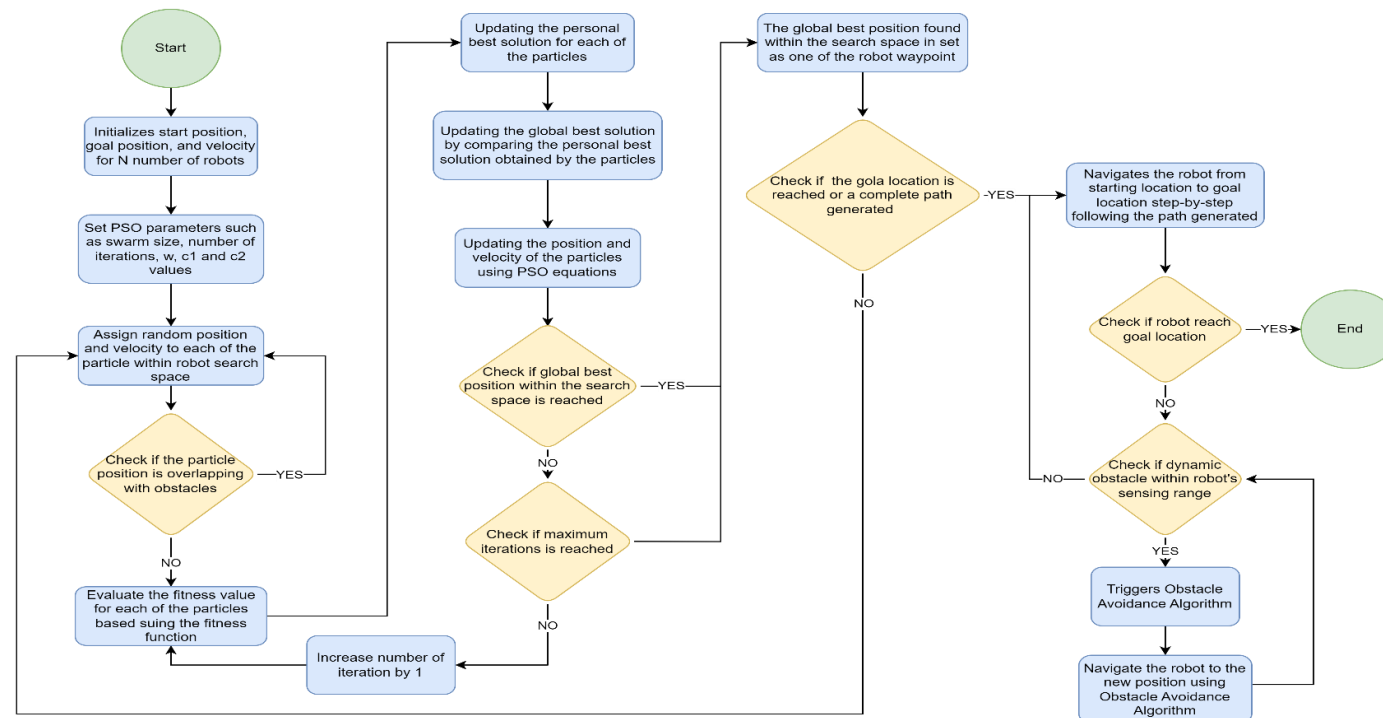


Figure 3.3: Flowchart PSO Path Planning Algorithm

3.2.6 Flowchart of Obstacle Avoidance Algorithm

Figure 3.4 illustrates the flow chart of Obstacle Avoidance Algorithm for a single robot. This flowchart will be implemented on all the robots in the MRS. The pseudocode for MPSO algorithm is provided in the Appendix PseudocodeA-2.

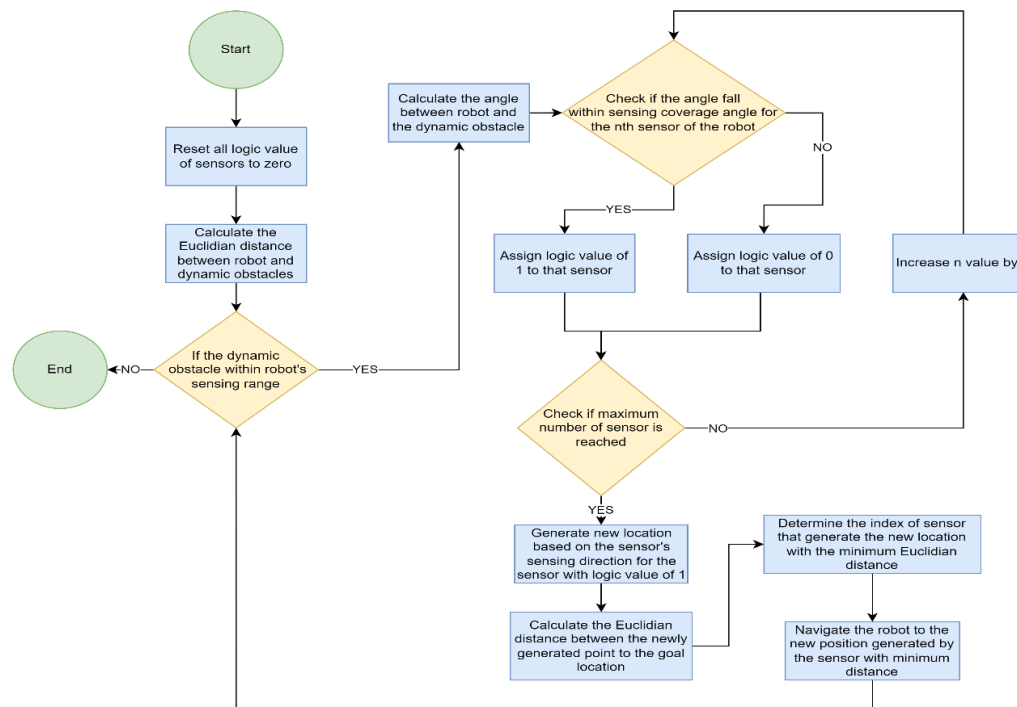


Figure 3.4: Flowchart for Obstacle Avoidance Algorithm

3.3 Computer Simulations

The multi-robot path planning will be carried out in three different scenarios as shown in Figure 3.5, Figure 3.6, and Figure 3.7 where three robots are initialized in different starting location, and they need to navigate through an environment filled with obstacles to reach their goal location respectively. These simulations will be conducted in MATLAB R2022a on a Windows laptop with 8th Generation Intel Core i5-8250U@1.6Ghz microprocessor with 8GB RAM. Each of the robots will consider other robots as dynamic obstacles. To simplify the simulation, some assumptions are to be made. For instance, due to the complex shape of a real robot and obstacles in real life, the simulation will be conducted with three robots in a 2-dimensional workspace where all the robots are enclosed with circle and all the obstacles are represented with simple shapes. Furthermore, the robot's movement is assumed to be holonomic, and the velocity of robot are assumed to be the same at 2 m/s. At the end of the simulation, the number of steps taken, trajectory path length and time required for each robot will be evaluated. The simulations are done with both basic PSO algorithm and MPSO algorithm to compare the performance of both algorithms.

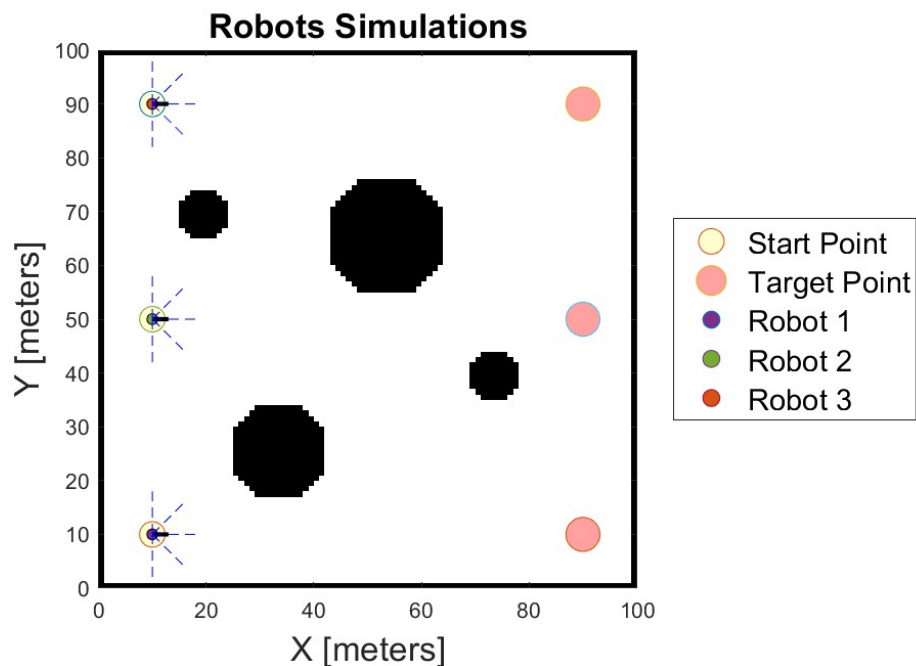


Figure 3.5: Scenario 1 (Simulation Environment with four Circular obstacles)

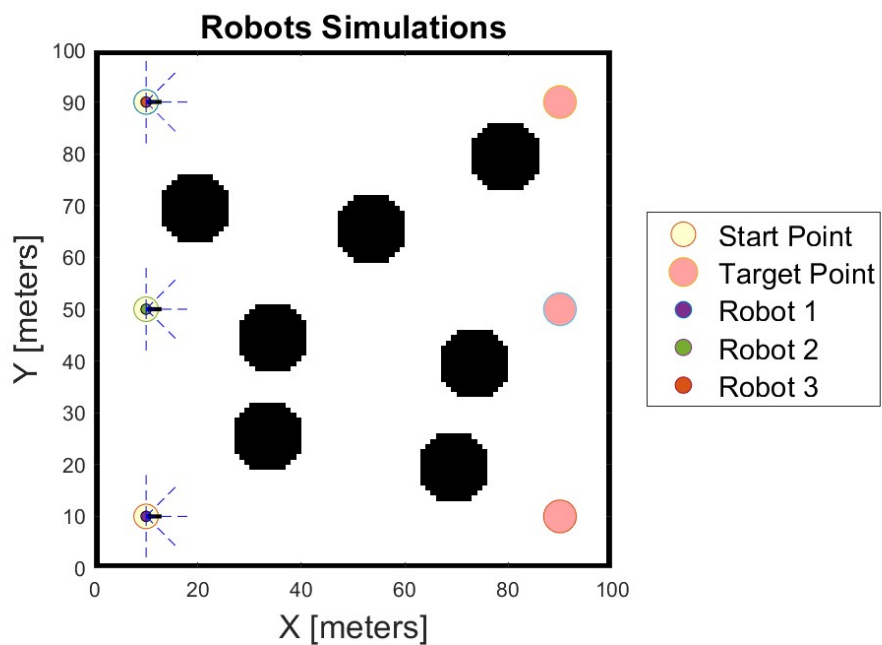


Figure 3.6: Scenario 2 (Simulation Environment with Seven Circular Obstacles)

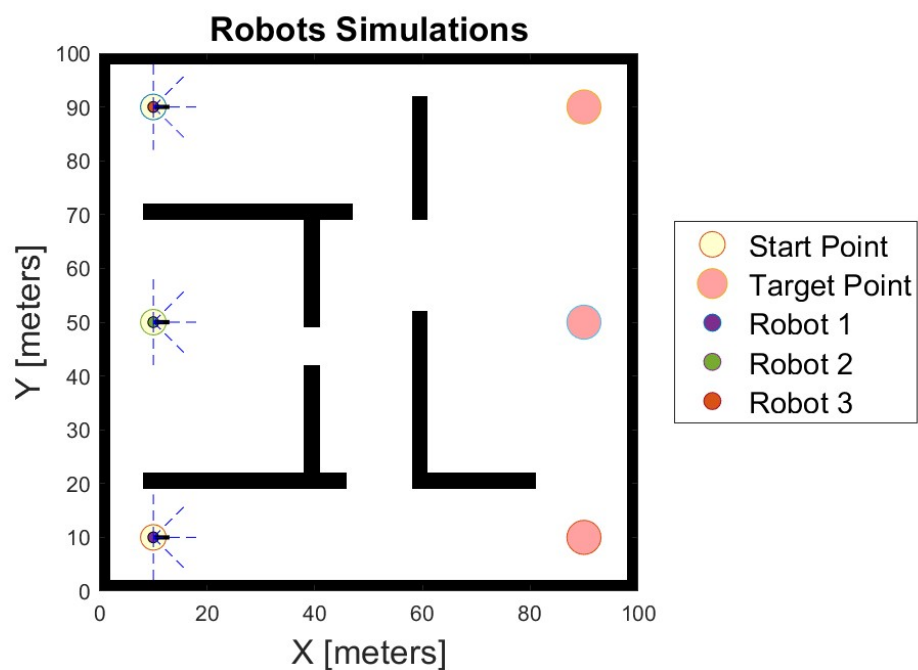


Figure 3.7: Scenario 3 (Simulation Environment with Multiple Vertical and Horizontal Obstacles)

3.4 Planning and Managing of Project Activities

This section describes the whole planning process in completing this 1-year project using two Gantt Charts as shown in Table 3.1 and Table 3.2. The project was divided into two parts, spanning two semesters, to aid in its successful completion. Besides that, for a better personal progress tracking, a more detailed Gantt chart was constructed for part I by using Gantt Project software. Both part I and part II of the project was done within 28 weeks in total. This project had not involved any cost because only software was used.

3.4.1 Project Part I

The part I of this project was conducted in 14 weeks, and it consists of four major activities to be accomplished. As shown in Table 3.1, the activities to be completed in part I were project formation and project planning, literature review, methodology research and progress report preparation. The first task of the whole project begins with project formation and project planning which was started at the beginning of project part I. Besides that, the background study was done on the FYP title to have general understanding on the title during the first two weeks. Not only that, the general Gantt chart for the whole project which includes the part I and part II of the project were prepared as shown in Table 3.1 and Table 3.2. Additionally, the more detail Gantt chart was also created as shown in Appendix GanttChartC-1 and GanttChartC-3 using a software named Gantt Project in the first three weeks of the project to have a better progress tracking for personal used. After that, a comprehensive literature reviews were conducted on the multi-robot path planning algorithms based on two main path planning approaches which are the classical approaches and heuristic approaches. In the classical approaches. RRT algorithm and APF algorithm were discussed by reviewing current papers published on single robot and multi-robot path planning. Besides. The heuristic algorithms such as ACO and PSO algorithm were also investigated. After the literature review, the different algorithm used in multi-robot path planning problem were compared and a suitable algorithm was chosen. Based on the chosen algorithm, the methodology for this project was designed and constructed. In-deep research on the chosen algorithm for deeper understanding on the mathematical models and a suitable platform,

programming language and robot for the simulation were also chosen. In this project, the simulations were mainly done with MATLAB. Besides that, a progress report was prepared to document all the research's results and the report preparation was started from week 3 together with literature review and it was submitted in week 14.

3.4.2 Project Part II

On the other hand, the part II of this project consists of five major tasks which are the PSO algorithm development and MATLAB simulation, result and discussion, conference paper preparation, poster preparation and final report preparation. The part II of this project focused on hands on development of the MRPP algorithm using MPSO in MATLAB software and the aim of this part is to investigate the performance of MRPP task in a simulated SAR operation in simulation. At the start of part II, the PSO algorithm was first developed in MATLAB software to evaluate the performance of the proposed algorithm in different scenarios such as the total time taken, the minimum path length, the safety factor and so on. After that, all the result gathered from the simulation were discussed, tabulated and documented in the report. At the same time, a conference paper was prepared and submitted for review using the result generated. At the end of part II, a poster was prepared for poster competition while at the same time, a final report was prepared to conclude the part I and part II of this project and this report was submitted in week 14 of part II. After that, this project was concluded with an oral presentation in week 14

Table 3.2: Gantt Chart (Part II)

No.	Activities	Week													
		1	2	3	4	5	6	7	8	9	10	11	12	13	14
1	MPSO Algorithm Development & MATLAB Simulation	✓	✓	✓	✓	✓	✓	✓							
2	Result & Discussion					✓	✓	✓	✓	✓	✓	✓			
3	Conference Paper Preparation							✓	✓	✓	✓				
4	Poster Preparation										✓	✓	✓	✓	
5	Final Report Preparation								✓	✓	✓	✓	✓	✓	✓

3.4.3 Summary

This chapter provides a detailed explanation of the MPSO algorithm, which is an improved version of the classical PSO algorithm. Unlike the PSO algorithm, the MPSO algorithm incorporates a new path planning scheme that determines each waypoint of the robot by running the entire MPSO algorithm, rather than determining one waypoint in each iteration in one run of the algorithm. Additionally, the inertial weight and learning factors of the MPSO algorithm are dynamically updated to balance the exploration and exploitation capability of the algorithm.

In this project, the combination of global path planning approach and local path planning approach is considered. MPSO algorithm is implemented as the global path planner to search for the complete path for all robots. To avoid obstacle during navigation, a sensor-based obstacle avoidance algorithm which known as Obstacle Avoidance Algorithm in this project is employed as a local path planner to help the robot reactively avoid any dynamic obstacles during navigation.

The performance for both PSO and MPSO algorithms were evaluated using MATLAB simulations in three different scenarios, with the same starting and goal locations for all scenarios, the settings of the scenarios are also discussed in this chapter. The first scenario is populated with four circular obstacles with different size, the second scenario is population with seven circular obstacles with the same size and the third scenarios is populated with multiple vertical and horizontal obstacles. The complexity of the robot's environment increases from the first scenario to the third scenario. All the scenarios were designed using binary occupancy map function provided in MATLAB. Furthermore, project planning and management were carefully executed, and Gantt chart were generated to ensure that all project activities were completed on time.

CHAPTER 4

RESULTS AND DISCUSSION

4.1 Introduction

This chapter focus on presenting and discussing the result obtained for all multi-robot scenarios using the proposed Modified Particle Swarm Optimization (MPSO) algorithm. This chapter involves the result comparison between PSO algorithm and MPSO algorithm in section 4.2 and the discussion on effect of PSO parameter on the simulation result in section 4.3.

4.2 Comparison Between PSO and MPSO Algorithms

In this section, the results obtained with the PSO algorithm and MPSO algorithms under all three scenarios are presented. Both algorithms were implemented with global path planning approach and the dynamic collision avoidance during the navigation of the robots was implemented using the Obstacle Avoidance Algorithm to help robot avoid any dynamic obstacles.

The basic PSO algorithm was implemented without the proposed path planning scheme. In PSO algorithm, the particles will be initialized at the robot starting positions, and the particles iteratively update their position and velocity using the velocity and position equations shown in equation $V_i(t+1) = \omega V_i(t) + c_1 \varphi_1(x_{PBest_i} - x_i(t)) + c_2 \varphi_2(x_{GBest} - x_i(t))$ (3.3) and $x_i(t+1) = x_i(t) + V_i(t+1)$ (3.2). In each of the iterations, one waypoint of the robot will be determined by evaluating the fitness value for each particle and determining the global best solution. Besides that, based on the velocity equation for the classical PSO in equation $V_i(t+1) = V_i(t) + c_1 \varphi_1(x_{PBest_i} - x_i(t)) + c_2 \varphi_2(x_{GBest} - x_i(t))$ (3.1), the inertial weight is not considered, and the social and cognitive learning factors are not dynamically updated instead they are set with a value of 2 throughout the whole simulation as the maximum cognitive and social learning factors used in this project is 2. When both cognitive and social learning factors are equal, they both exert an equal level of influence on the algorithm's performance. On the other hand, the MPSO algorithm implemented with the new path planning

scheme where the particles are first initialized within a predefined search space, and they iteratively search for a global best position within the search space. The algorithm stops once the global best position is determined and this global best position will be set as one of the waypoints of the robot. A new MPSO algorithm is then reiterated again by first initialising the particles within a predefined search space around the previous generated waypoint and determine the next waypoint by searching for the global best position within the search space. These processes are repeated until a complete path is generated.

In order to compare the performance of both algorithms, both algorithms were simulated with five times for each scenario, and they were evaluated in terms of average path length and execution time using the similar set of PSO parameters as shown in Table 4.1. Besides that, all three scenarios having the same dimension which is 100 meters x 100 meters and the robots in all three scenarios are having their own respective starting and goal position as shown in Table 4.2.

Table 4.1: PSO Parameters

Parameters	Value
Swarm Size	100
Maximum Iteration, $iter_{total}$	30
Maximum Inertial Weight, ω_{max}	0.95
Minimum Inertial Weight, ω_{min}	0.4
Maximum Cognitive Parameter, c_{1max}	2.0
Minimum Cognitive Parameter, c_{1min}	0.5
Maximum Social Parameter, c_{2max}	2.0
Minimum Social Parameter, c_{2min}	0.5
Particle's Maximum Velocity, V_{max}	8
Particle's Minimum Velocity, V_{min}	2

Table 4.2: Starting and Goal Position for Each Robot

	Starting Points (x, y)	Target Points (x, y)

Robot 1	10	10	90	90
Robot 2	10	50	90	50
Robot 3	10	90	90	10

4.2.1 Scenario 1 (Four Circular Obstacles)

In the first scenario as shown in Figure 3.5, four static obstacles are considered (same shape, different size). Each robot is starting from different starting position and targeted to reach different goal position.

4.2.1.1 PSO Algorithm Result

Figure 4.1 illustrates the particles position and the robot's waypoints generated in each iteration. Each coloured point represents a particle and the particles with the same coloured represents the particles in the same iteration.

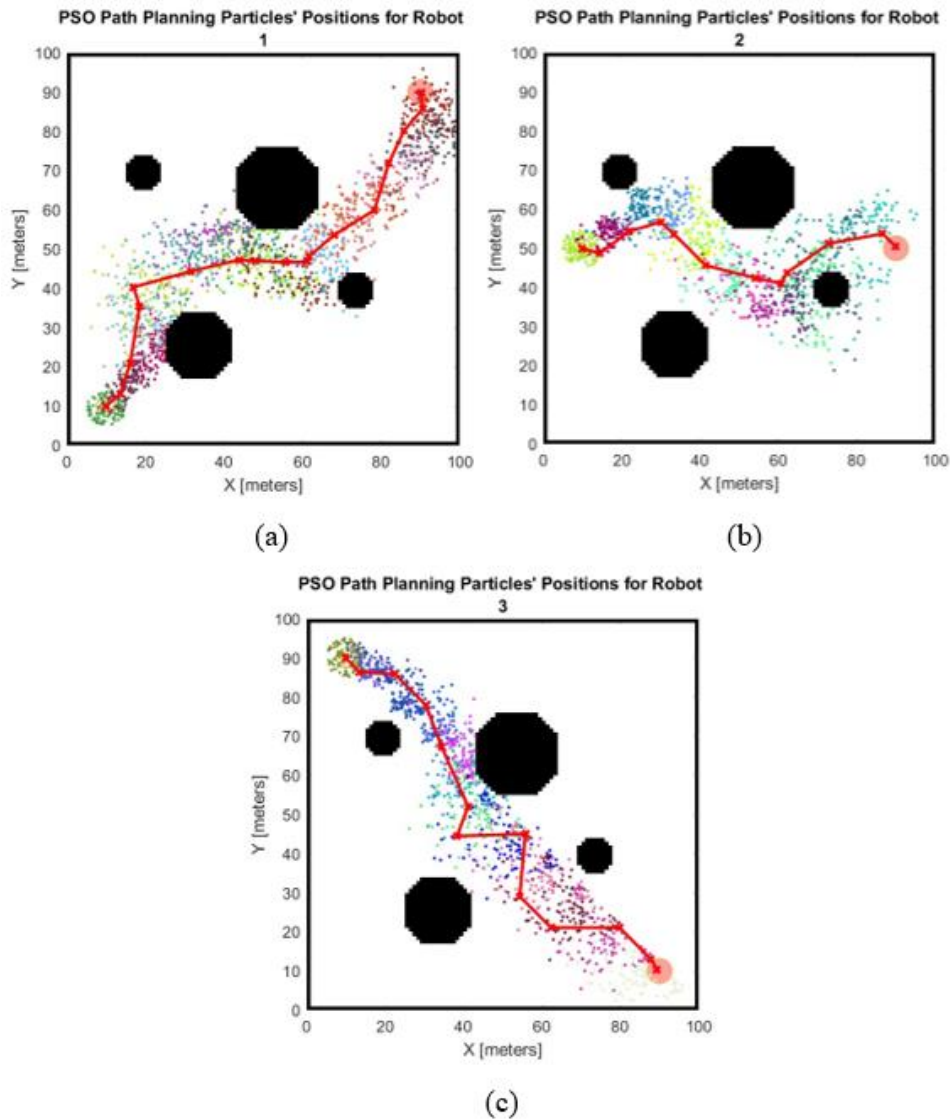


Figure 4.1: Particles' Position and Waypoints in Scenario 1 With PSO

Algorithm: (a) Robot 1; (b) Robot 2; (c) Robot 3

After each robot path is generated using the PSO algorithm, the robots start navigating toward the target position following the generated path. The trajectory for each of the robot from respective starting position to goal position are shown in Figure 4.2. During the navigation of the robots, robot 1 and robot 3 sensed each other with their sensor in simulation step 5 as shown in Figure 4.3. Once the robot sensed any dynamic obstacle in its sensing range, the robot triggers the Obstacle Avoidance Algorithm in order to help the robot adjust its navigation and avoid colliding with other robot as shown in simulation step 8 in Figure 4.4. From the simulation, both of the robots successfully avoid collision with the help of the Obstacle Avoidance

Algorithm. To have a better understanding on the simulation, the whole simulation result is provided in the Appendix SimulationB-1.

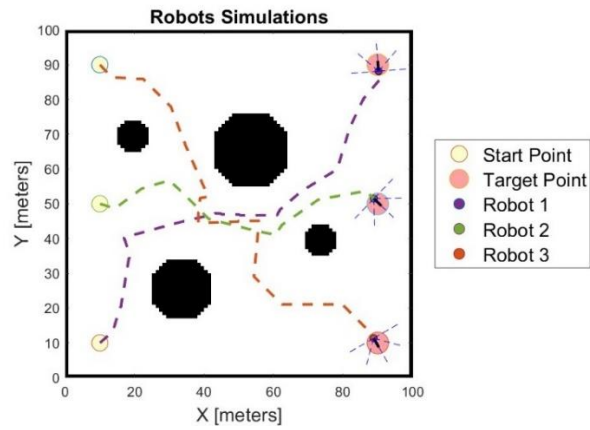


Figure 4.2: Trajectory of Robots in Scenario 1 With PSO Algorithm

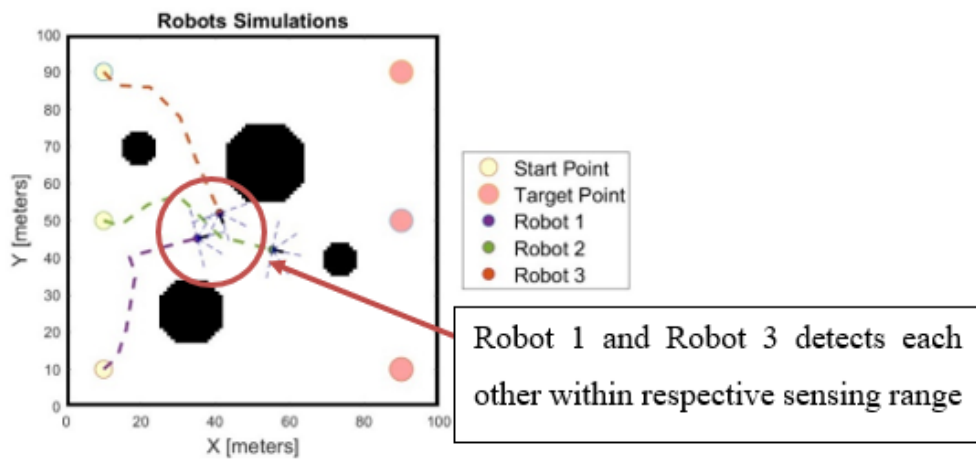


Figure 4.3: Dynamic Obstacle Detection of Robot 1 and Robot 3 in Scenario 1 (Simulation Step 5)

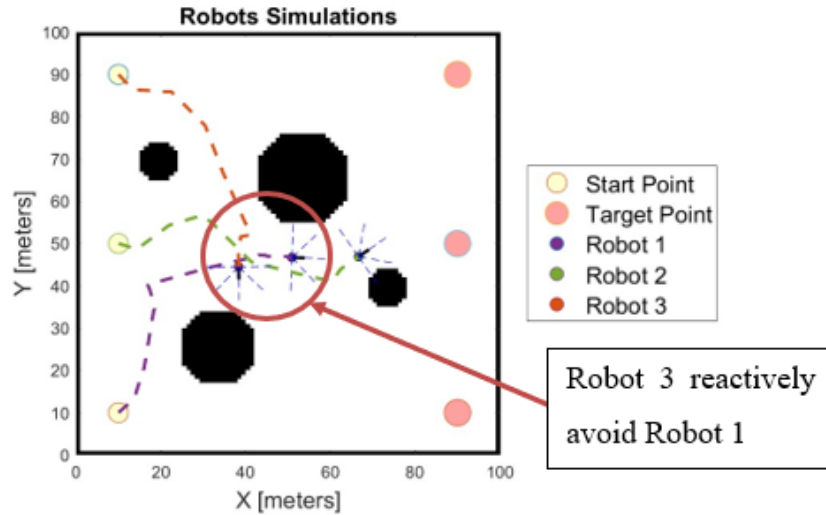


Figure 4.4: Collision Avoidance Algorithm Triggered to Avoid Collision in Scenario 1 (Simulation Step 8)

Moreover, it is possible to calculate the total best fitness of all three robots in each iteration using equation $F = \lambda_1 F_1 + \lambda_2 F_2$ (3.9), and a graph displaying the global best fitness over the course of the iterations can be generated as illustrated in Figure 4.5. According to the plotted graph, it illustrates that the path for all robots can be obtained after 16 iterations, and it shows a gradual decrease in fitness values from iteration 1 until iteration 16. Furthermore, the maximum global best fitness was achieved in iteration 1 with a value of 339.61, while the minimum global best fitness was attained in iteration 16 with a value of 32.33 as shown in Table 4.3 and Figure 4.5.

Table 4.3: Global Best Fitness of Each Iteration in Scenario 1 With PSO

No of Iterations	Global Best Fitness
1	339.61
2	315.65
3	297.90
4	271.08
5	250.32
6	217.23
7	188.35
8	170.71

9	155.51
10	129.00
11	96.64
12	77.32
13	61.06
14	47.16
15	36.70
16	32.33

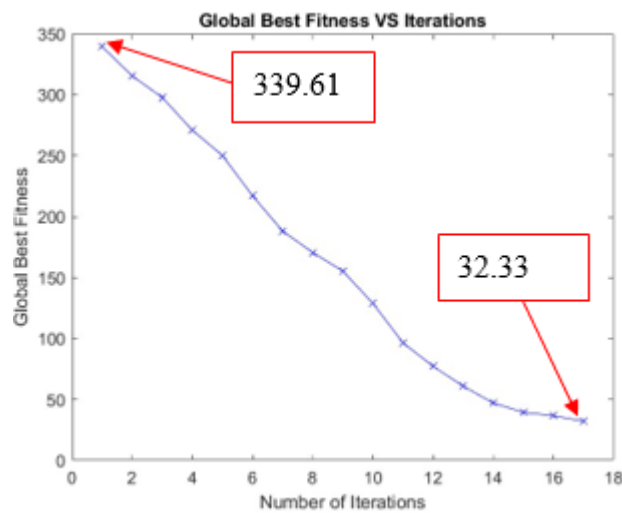


Figure 4.5: Graph of Global Best Fitness vs Iteration for Scenario 1 With PSO Algorithm

4.2.1.2 MPSO Algorithm Results

Figure 4.6 illustrates the waypoints for each of the robot generated with MPSO algorithm. At the same time, the position of the particles in the last run of MPSO algorithm is shown in the figure. The particles with the same colour represent the particle in the same iteration. From the figure, we can observe that the particles were initialized within a predefined search space around the previous generated waypoint. In order to determine the next waypoint, the particles iteratively update its velocity and position with equations $V_i(t+1) = \omega V_i(t) + c_1 \varphi_1(x_{PBest_i} - x_i(t)) + c_2 \varphi_2(x_{GBest} - x_i(t))$ (3.3) and $x_i(t+1) = x_i(t) + V_i(t+1)$ (3.2) to search for the global best

position within the search space. The algorithm was terminated when the global best position is no longer updated.

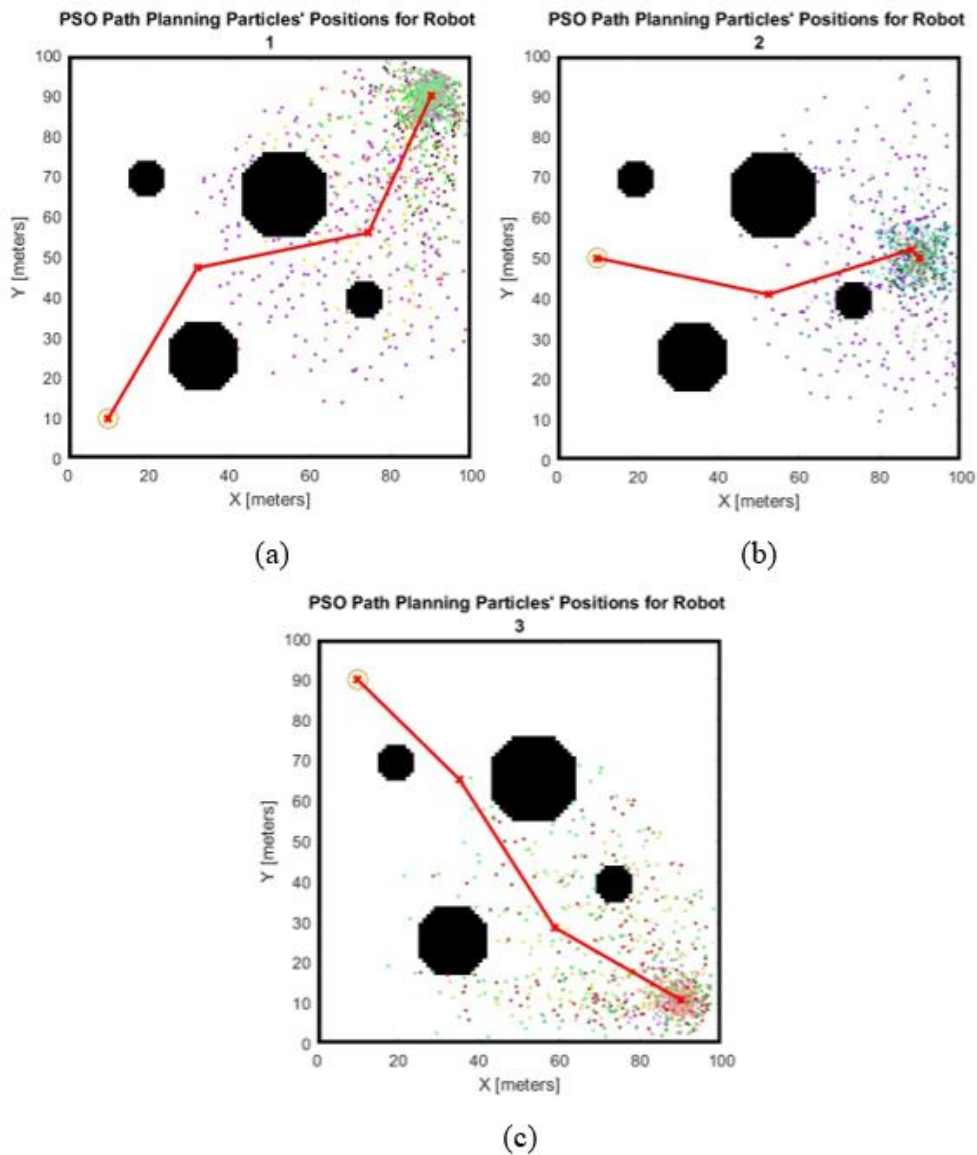


Figure 4.6: Particles' Position and Waypoints in Scenario 1 With MPSO

Algorithm: (a) Robot 1; (b) Robot 2; (c) Robot 3

The trajectory generated for each robot with MPSO algorithm is shown in Figure 4.7. Based on the figure, it can be observed that the path planned by the algorithm successfully avoid collision with any obstacle. The detailed simulation process can be observed in Appendix SimulationB-2 for deeper understanding.

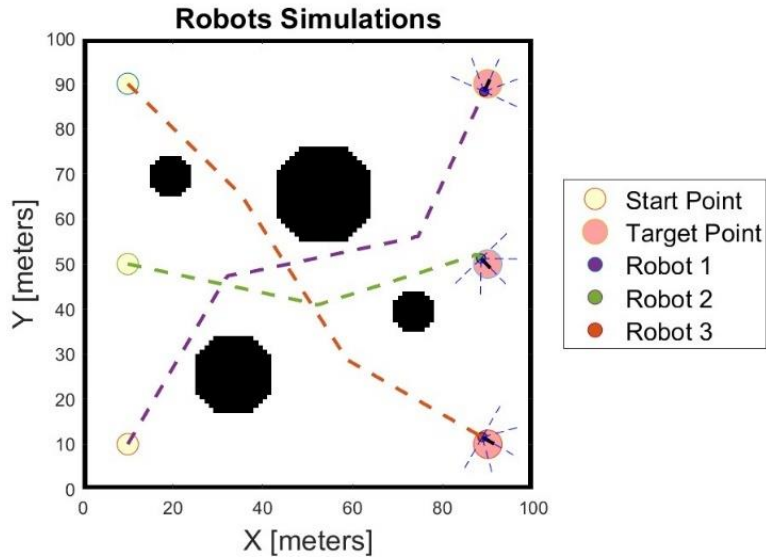


Figure 4.7: Trajectory of Robots in Scenario 1 With MPSO Algorithm

Similarly, the total best fitness for all three robots in each iteration can be calculated and a graph displaying the global best fitness over the course of the iterations is shown in Figure 4.8. It can be observed that only four iterations are required to compute all robots' path. Besides that, the maximum global best fitness found in iteration 1 is 339.61 while the minimum global best fitness found in iteration 4 is 31.05 as illustrated in Figure 4.8 and Table 4.4.

After running through the simulation with both PSO and MPSO algorithms for five times, the performance of both algorithms in scenario 1 can be evaluated and tabulated in terms of average path length and execution time as shown in Table 4.5.

Table 4.4: Global Best Fitness of Each Iteration in Scenario 1 With MPSO

No of Iterations	Global Best Fitness
1	339.61
2	214.04
3	100.49
4	31.05

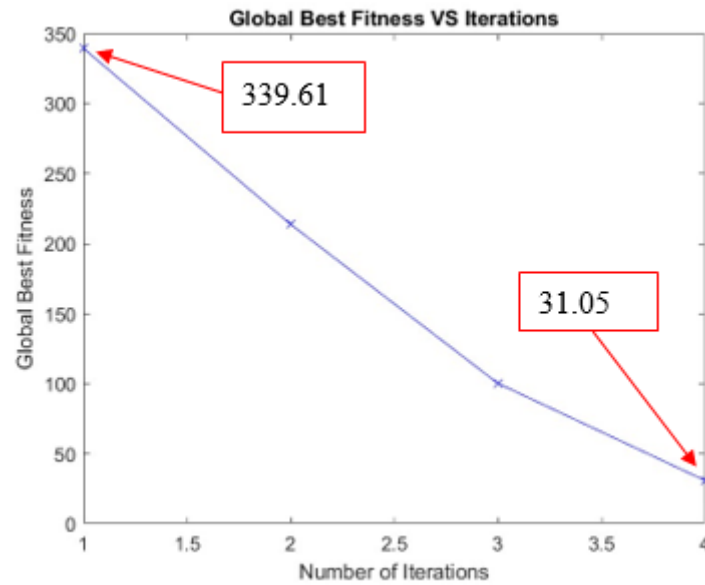


Figure 4.8: Graph of Global Best Fitness vs Iteration for Scenario 1 With MPSO Algorithm

Table 4.5: Result Obtained with PSO and MPSO Algorithms in Scenario 1

	Start Point	Target Point	Total Number of Iterations		Average Path Length (meters)		Average Execution Time (s)	
			PSO	MPSO	PSO	MPSO	PSO	MPSO
Robot 1	(10,10)	(90,90)	16	4	131.86	122.28	98.46	96.43
Robot 2	(10,50)	(90,50)			91.84	81.69	68.52	62.36
Robot 3	(10,90)	(90,10)			138.94	113.44	96.97	83.59

Based on the result in Table 4.5, it can be observed that the average path length and the average execution time obtained by MPSO for all three robots is shorter compared to that of the PSO algorithm. Thus, it can be said that the MPSO algorithm outperform the PSO algorithm in terms of average path length and execution time in scenario 1.

4.2.2 Scenario 2 (Seven Circular Obstacles)

In the second scenario as shown in Figure 3.6, the number of static obstacles increased from three to seven where the environment is populated with seven obstacles (same shape, same size). This is to evaluate the performance of the algorithms in an environment pack with obstacles.

4.2.2.1 PSO Algorithm Results

Figure 4.9 illustrates the particles position and the robot's waypoints generated in each iteration in scenario 2 using the PSO algorithm.

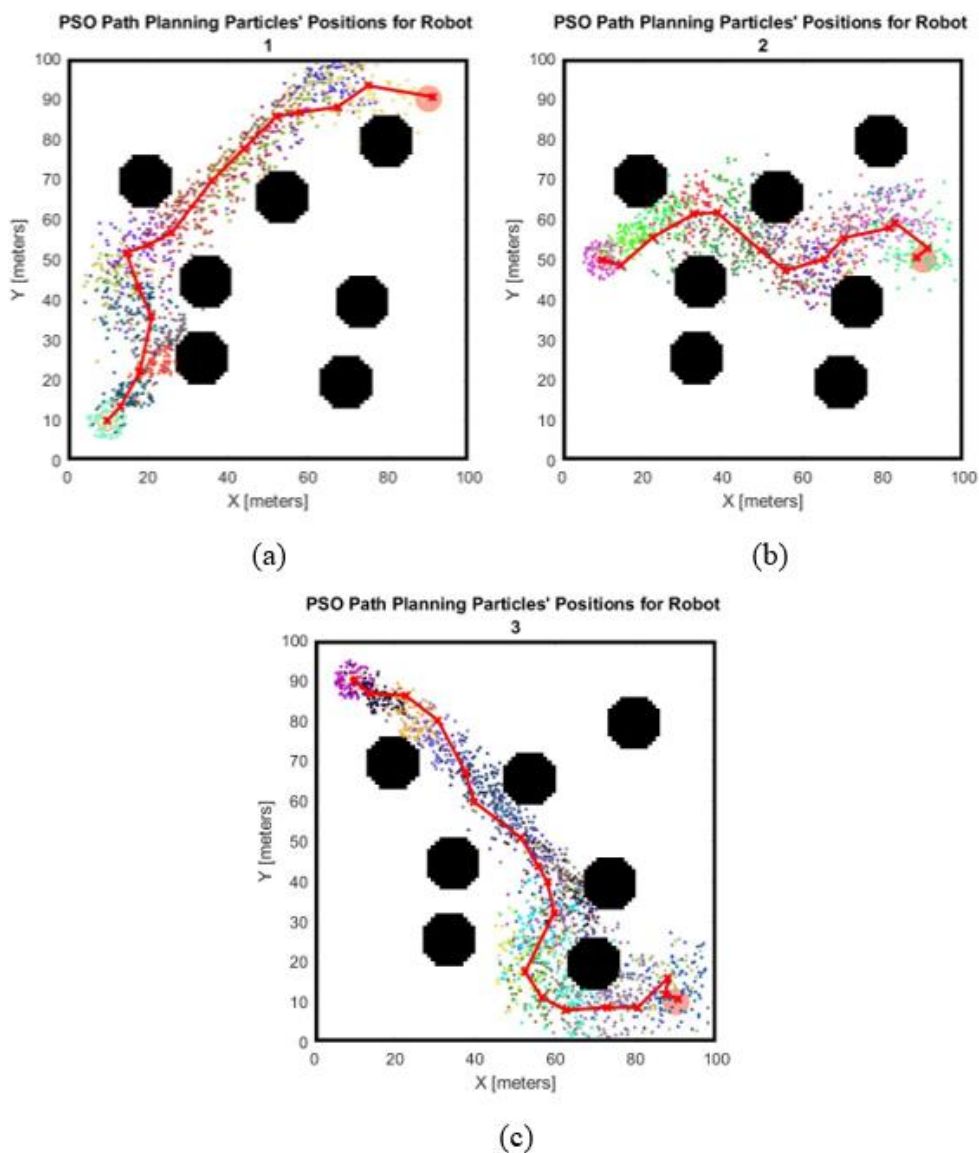


Figure 4.9: Particles' Position and Waypoints in Scenario 2 With PSO Algorithm: (a) Robot 1; (b) Robot 2; (c) Robot 3

Besides that, the trajectory generated for all robots in scenario 2 using PSO algorithm can be observed in Figure 4.10. From the simulation result obtained in Figure 4.10 and the Appendix SimulationB-3, it can be observed that the path planned by the algorithm successfully avoid collision with any obstacle.

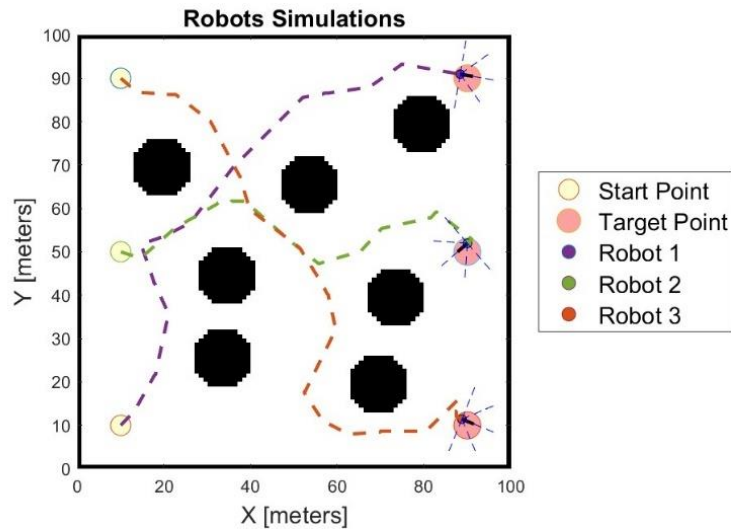


Figure 4.10: Trajectory of Robots in Scenario 2 With PSO Algorithm

Moreover, the graph of global best fitness over the course of iterations are displayed in Figure 4.11. From the graph, it can be observed that the total number of iterations required to generate all robots' path are 18 iterations. The maximum global best fitness in iteration 1 is 339.61 while the minimum global best fitness in iteration 18 is 34.45 as shown in Table 4.6 and Figure 4.11.

Table 4.6: Global Best Fitness of Each Iteration in Scenario 2 With PSO

No of Iterations	Global Best Fitness
1	339.61
2	316.49
3	299.83
4	271.58
5	245.44
6	227.54

7	198.07
8	180.09
9	152.50
10	126.31
11	112.19
12	100.90
13	89.50
14	82.74
15	59.02
16	45.92
17	37.87
18	34.45

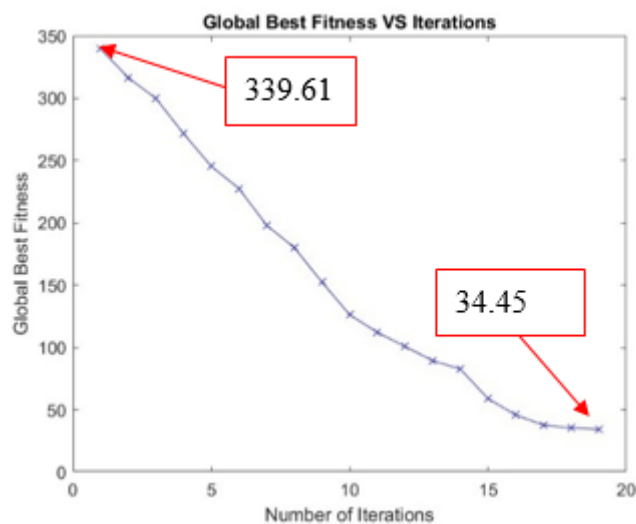


Figure 4.11: Graph of Global Best Fitness vs Iteration for Scenario 2 With PSO Algorithm

4.2.2.2 MPSO Algorithm Result

The waypoints generated by the MPSO algorithm for scenario 2 are depicted in Figure 4.12. From the finding illustrated in Figure 4.12 (c), it is noticeable that Robot 3 successfully created a path between two obstacles that were located closely to each other, leading to a reduced overall path length. In comparison to the trajectory produced by the PSO and MPSO algorithms for Robot 3, the PSO algorithm failed to generate a path between obstacles with

narrow gap as shown in Figure 4.9 (c) which result in an increased overall path length.

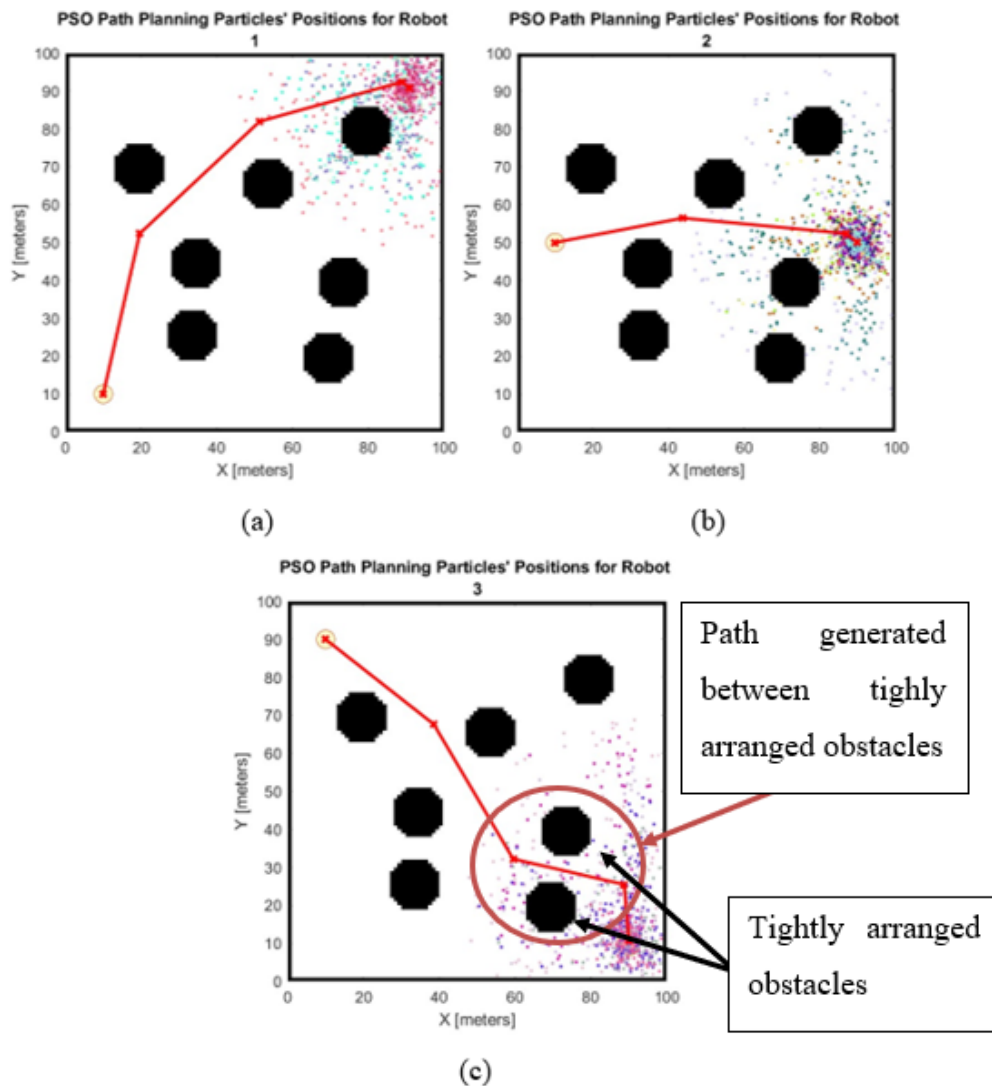


Figure 4.12: Particles' Position and Waypoints in Scenario 2 With MPSO

Algorithm: (a) Robot 1; (b) Robot 2; (c) Robot

The trajectory of the robots is illustrated in Figure 4.13. From the result, the Obstacle Avoidance Algorithm was not triggered throughout the entire navigation process as the robots did not detect any dynamic obstacles within its sensing range. The entire simulation process of the algorithm in scenario 2 is illustrated in Appendix SimulationB-4 for deeper understanding.

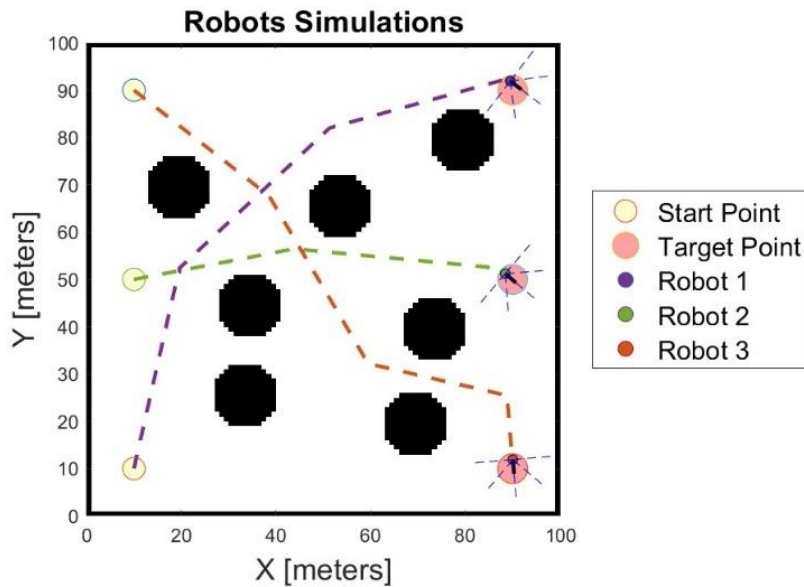


Figure 4.13: Trajectory of Robots in Scenario 2 With MPSO Algorithm

Furthermore, the graph of global best fitness over the course of iterations for scenario 2 with MPSO algorithm is displayed in Figure 4.14. The entire MPSO algorithm was simulated for a total of five times repeatedly to generate the path for all robots. Besides, it can be observed that the maximum global best fitness found in iteration 1 is 339.61 while the minimum global best fitness found in iteration 5 is 33.31 as shown in Table 4.7 and Figure 4.14.

After running the simulations for both PSO and MSPO algorithms in scenario 2 for five times, the performance for both algorithms are evaluated and compared in terms of average path length and execution time. The results are tabulated in Table 4.8.

Table 4.7: Global Best Fitness of Each Iteration in Scenario 2 With MPSO

No of Iterations	Global Best Fitness
1	339.61
2	237.11
3	109.86
4	50.26
5	33.31

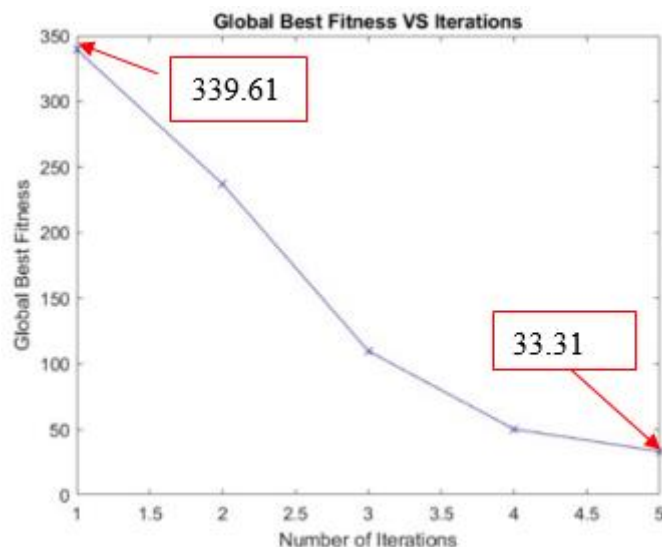


Figure 4.14: Graph of Global Best Fitness vs Iteration for Scenario 2 With MPSO Algorithm

Table 4.8: Result Obtained with PSO and MPSO Algorithms in Scenario 2

	Start Point	Target Point	Total Number of Iterations		Average Path Length (meters)		Average Execution Time (s)	
			PSO	MPSO	PSO	MPSO	PSO	MPSO
Robot 1	(10,10)	(90,90)	18	5	136.16	126.76	91.87	90.31
Robot 2	(10,50)	(90,50)			98.40	79.83	70.22	65.21
Robot 3	(10,90)	(90,10)			147.11	121.41	103.13	91.28

The data presented in Table 4.8 indicates that the MPSO algorithm performs better than the PSO algorithm in scenario 2 in regard to average path length and execution time. As the PSO algorithm was unable to generate a path between two obstacles with narrow gap, the path length produced for Robot 3 with the PSO algorithm was approximately 25 meters longer than the path generated by the MPSO algorithm. Therefore, the MPSO algorithm's capability to find a path through closely arranged obstacles enabled it to produce a shorter path.

4.2.3 Scenario 3 (Multiple Long walls)

In the third scenario as shown in Figure 3.7, the algorithms were tested in an environment populated with multiple vertical and horizontal obstacles to evaluate the performance of the algorithms in environment populated with vertical and horizontal obstacles.

4.2.3.1 PSO Algorithm Results

Figure 4.15 illustrates the particles' position and the robot's waypoints for each iteration of PSO algorithm in scenario 3.

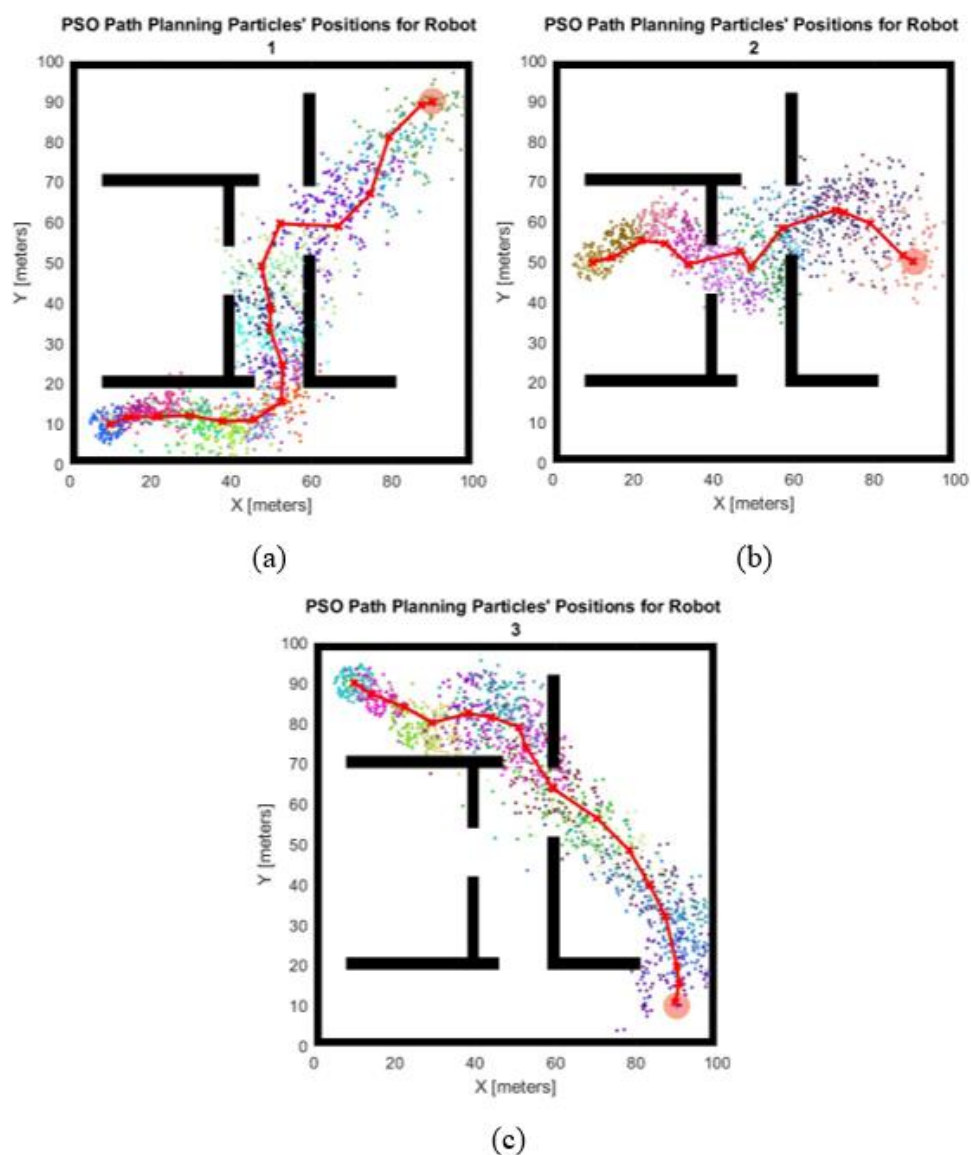


Figure 4.15: Particles' Position and Waypoints in Scenario 3 With PSO Algorithm: (a) Robot 1; (b) Robot 2; (c) Robot 3

The trajectory for all the robots in scenario 3 can be observed in Figure 4.16. During the robots' navigation, Robot 2 and Robot 3 detects each other at simulation step 6 as shown in Figure 4.17. Once the robot detects any dynamic obstacle within its sensing range. The robot, subsequently, will avoid the dynamic obstacle by activating the Obstacle Avoidance Algorithm as shown in Figure 4.18. The result illustrates that the collision between Robot 2 and Robot 3 is successfully avoided. The detailed simulation processes are illustrated in the Appendix SimulationB-5.

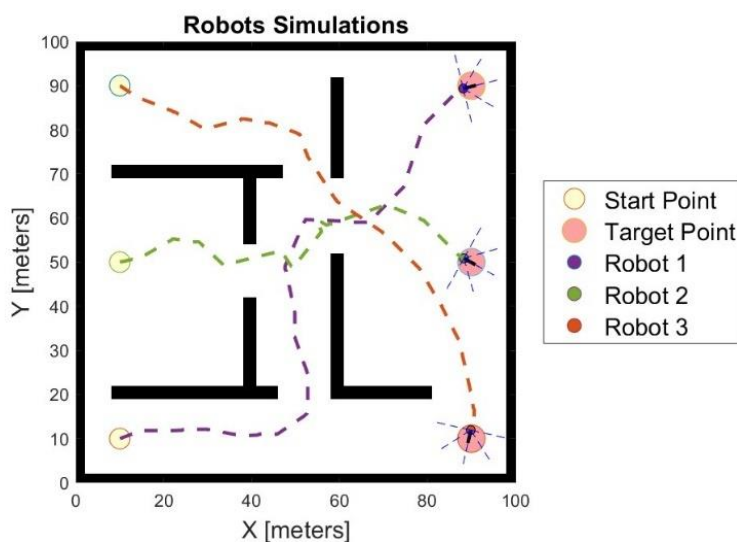


Figure 4.16: Trajectory of Robots in Scenario 3 With PSO Algorithm

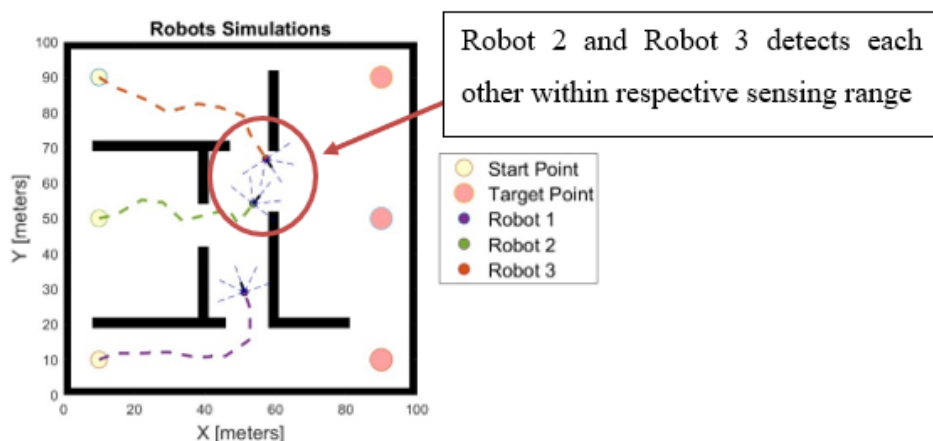


Figure 4.17: Dynamic Obstacle Detection of Robot 2 and Robot 3 in Scenario 3 (Simulation Step 6)

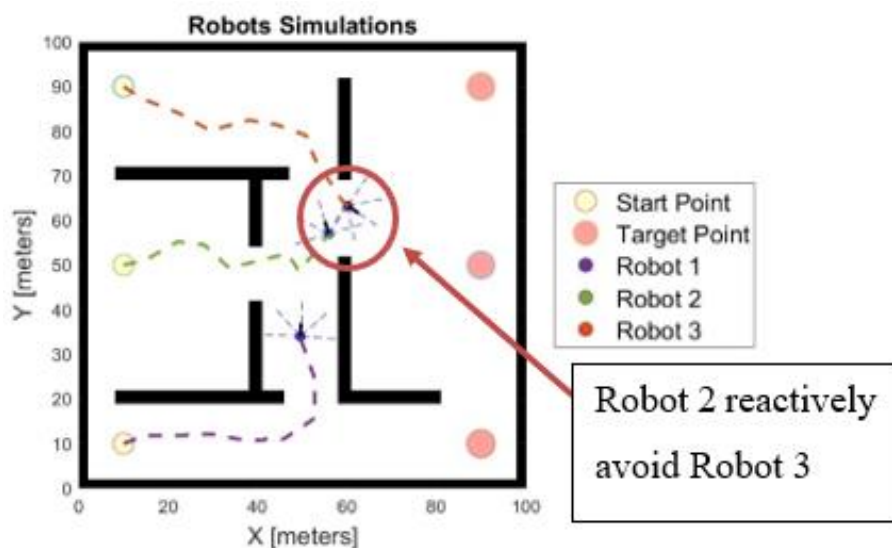


Figure 4.18: Collision Avoidance Algorithm Triggered to Avoid Collision in Scenario 3 (Simulation Step 7)

The graph of global best fitness over the course of iterations in scenario 3 using PSO algorithm is illustrated in Figure 4.19. It can be observed that the total number of iterations required to compute all robots' path is 16 iterations. The maximum global best fitness found in iteration 1 is 339.61 while the minimum global best fitness found in iteration 16 is 34.75 as shown in Table 4.9 and Figure 4.19.

Table 4.9: Global Best Fitness of Each Iteration in Scenario 3 With PSO

No of Iterations	Global Best Fitness
1	339.61
2	320.81
3	299.18
4	287.35
5	274.25
6	256.68
7	238.85
8	231.13
9	191.23
10	165.59

11	135.90
12	106.89
13	91.69
14	66.87
15	48.11
16	36.34
17	34.75

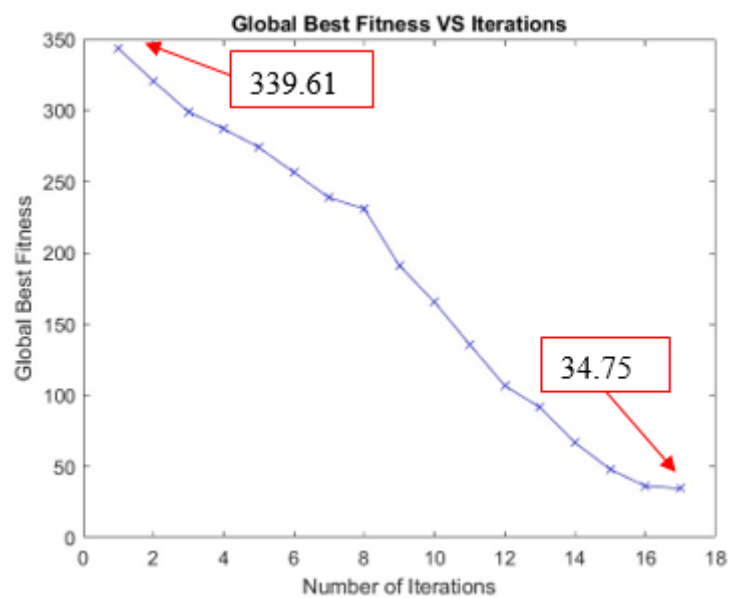


Figure 4.19: Graph of Global Best Fitness vs Iteration for Scenario 3 With PSO Algorithm

4.2.3.2 MPSO Algorithm Results

Figure 4.20 illustrates the robot waypoints generated with MPSO algorithm in scenario 3.

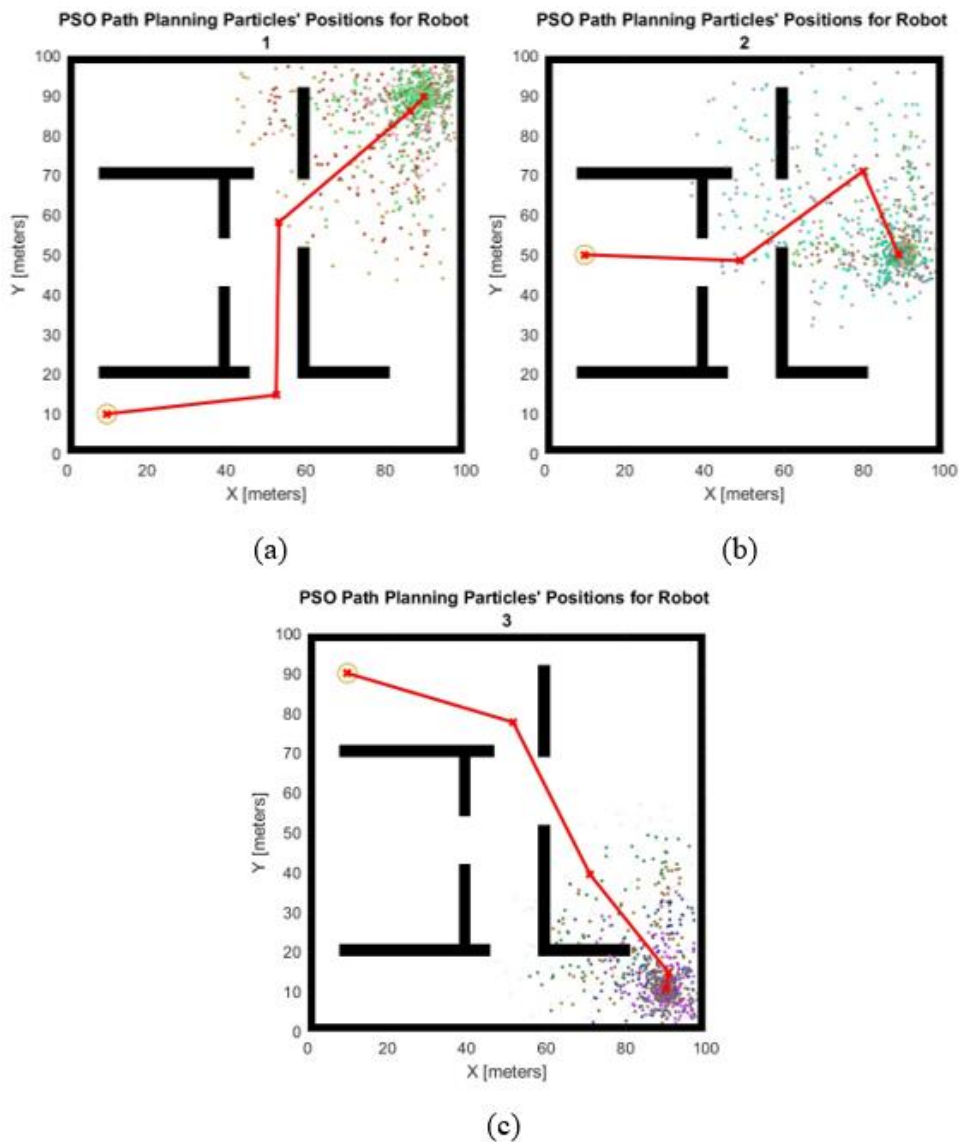


Figure 4.20: Particles' Position and Waypoints in Scenario 3 With MPSO Algorithm: (a) Robot 1; (b) Robot 2; (c) Robot 3

The trajectory of all robots in scenario 3 using MPSO algorithm is illustrated in Figure 4.21. From the result in Figure 4.21 and Appendix SimulationB-6, the path planned by the algorithm successfully avoid collision with any obstacle.

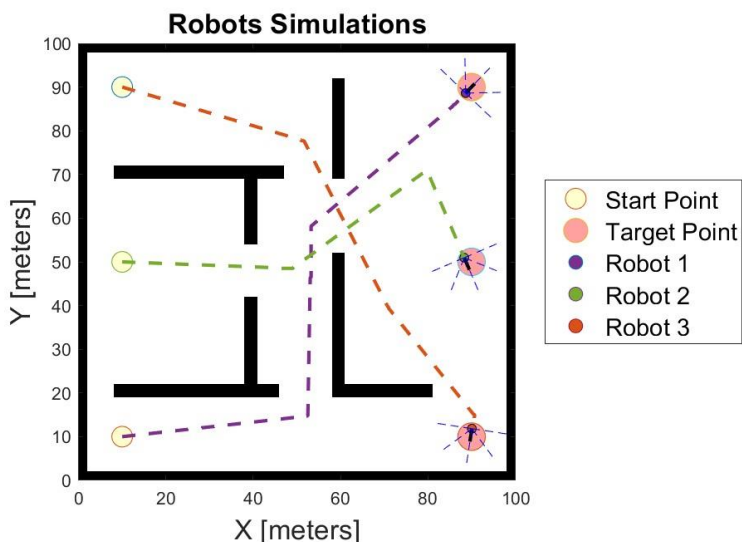


Figure 4.21: Trajectory of Robots in Scenario 3 With MPSO Algorithm

Error! Reference source not found. shows the plot of global best fitness against the number of iterations. It is observed that the calculation of paths for all robots requires five iterations. The global best fitness value decreases quickly from iteration 1 to iteration 4, and the decrement from iteration 4 to iteration 5 is very minimal as Robot 1 and Robot 2 have already reached their target position in iteration 4 and the global best fitness for Robot 1 and Robot 2 are already at the minimal value. The maximum global best fitness found in iteration 1 is 339.61 while the minimum global best fitness found in iteration 5 is 34.75 as shown in Table 4.10 and **Error! Reference source not found.**

Upon conducting the simulations five times each for PSO and MPSO algorithms in scenario 3, their performance was evaluated and compared in terms of average path length and execution time as shown in Table 4.11

Table 4.10: Global Best Fitness of Each Iteration in Scenario 3 With MPSO

No of Iterations	Global Best Fitness
1	339.61
2	238.63
3	134.75
4	40.18
5	34.34

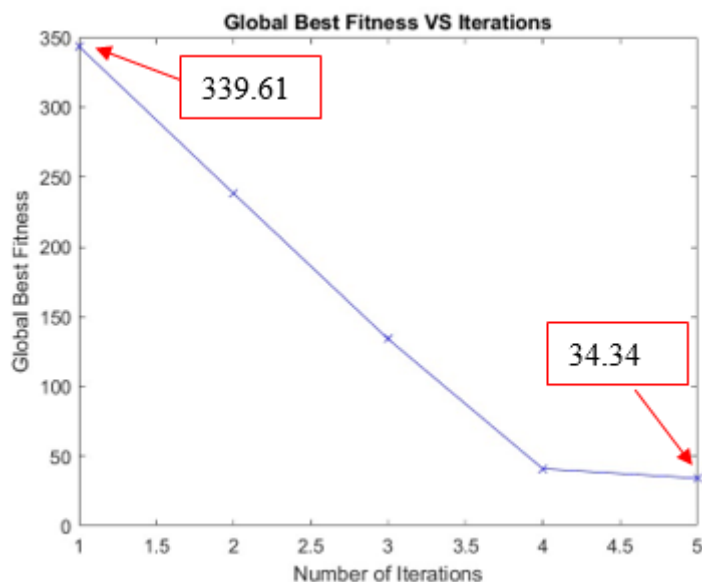


Figure 4.22: Graph of Global Best Fitness vs Iteration for Scenario 3 With MPSO Algorithm

Table 4.11: Results Obtained with PSO and MPSO Algorithms in Scenario 3

	Start Point	Target Point	Total Number of Iterations		Average Path Length (meters)		Average Execution Time (s)	
			PSO	MPSO	PSO	MPSO	PSO	MPSO
Robot 1	(10,10)	(90,90)	16	5	144.03	132.50	107.1	95.95
Robot 2	(10,50)	(90,50)			98.00	96.45	81.28	67.73
Robot 3	(10,90)	(90,10)			126.2	120.74	89.90	89.18

The findings presented in Table 4.11 show that, in scenario 3, the MPSO algorithm continues to outperform the PSO algorithm in terms of average path length and execution time. While the average path lengths and execution times for Robot 2 and Robot 3 are comparable between the two algorithms, the path generated with MPSO is noticeably straighter than the path generated with PSO algorithm. This implies that the robot will have to

make fewer rotations to alter its direction during the navigation, leading to reduced energy consumption.

Upon analysing and comparing the performance of both PSO and MSPO algorithms across all three scenarios, it can be inferred that the MPSO algorithm proves to be more effective and efficient than the PSO algorithm. Thus, the MPSO algorithm can be deemed superior to the PSO algorithm in all three scenarios.

4.3 MPSO Parameters

This section discusses the effect of PSO parameter on the simulation result. There are several parameters that need to be tuned for better performance in multi-robot path planning such as population size, inertial weight, cognitive learning factor, and social learning factor. The choice of PSO parameter can significantly affect the performance of the multi-robot path planning algorithm, thus, the parameters need to be chosen wisely. In order to find the suitable PSO parameter, scenario 1 as shown in Figure 4.23 is used to evaluate the effect of the parameters on the simulation result.

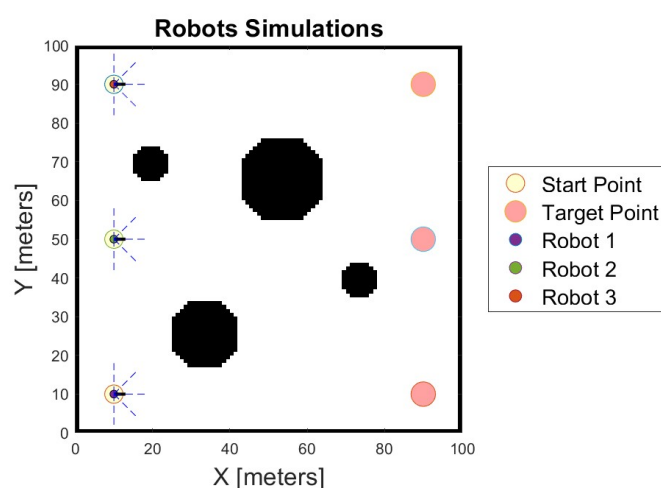


Figure 4.23: Scenario 1 Used to Evaluate the Effect of PSO Parameters

4.3.1 Population Size

In multi-robot path planning using PSO algorithm, population size or the particle swarm size is an important parameter that affects the performance of the algorithm. Population size refers to the number of particles or number of candidate solution present in a swarm that are generated and evaluated in each iteration of the algorithm. Since each of the particle in the swarm represents a candidate solution, increasing the population size able to improve the performance of the PSO by increasing the exploration capability of the algorithm when searching for solution. At the same time, the particles will also have lower chances in getting stuck in local optimum. However, even though the performance and solution might get improved with increment in population size, the computation cost of the algorithm can also increase significantly as the particles will need to be evaluated in every iteration. On the other hand,

decreasing the population size can result faster execution time but it decreases the exploration capability of the algorithm in searching for solution which may result in premature convergence and lower quality solution. In the worst-case scenario, the particles might get stuck in local optimum and fail to find the global optimum solution. Therefore, the choice of population size in multi-robot path planning using PSO involves a trade-off between convergence speed and solution quality, as well as the computational resource available for the simulation. Careful tuning of the population size is necessary to balance the convergence speed and the solution quality in order to achieve the best performance of the PSO algorithm.

The effect of the population size on the simulation result is tested by comparing the result obtained with different population size of 20, 50, 100, and 200. The simulation is repeated for each population size and the effect of the population size is evaluated in terms of the average path length and average execution time of all robots.

Table 4.12: Simulation results with different population sizes

Population Size	Average path lengths (m)	Average execution time (s)
20	113.22	53.07
50	109.84	59.04
75	107.75	59.42
100	107.80	71.27
200	107.83	108.23

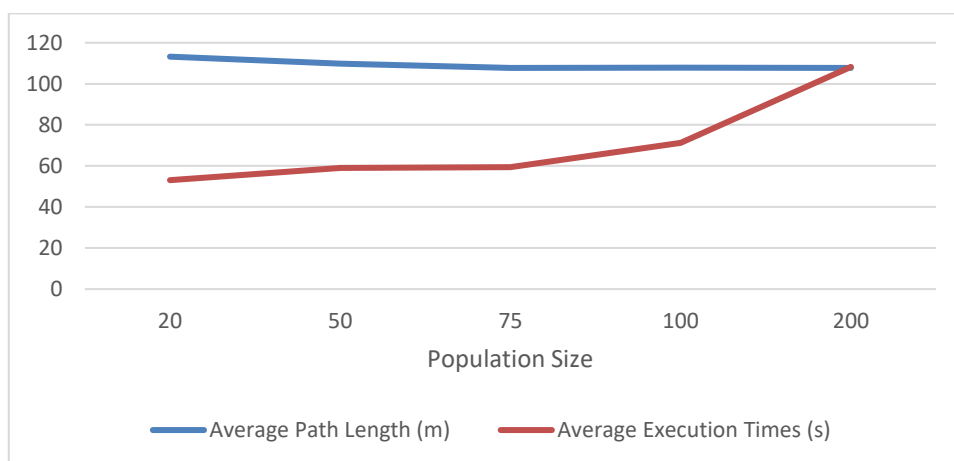


Figure 4.24: Simulation Results Obtained for Different Population Size

The result presented in Table 4.12 and Figure 4.24 indicate that as the population size increases from 20 to 200, there is a gradual decrease in the average path length, from 113.22m to 107.83m. Notably, the average path length obtained with a population size of 75 is very close to that obtained with a population size of 100 and 200, indicating that further improvement in path length may be unlikely. However, as the population size increases, the average execution time increases substantially. Specifically, the results show that when a population size of 100 or 200 is used, the execution time increases from 71.27 seconds to 108.23 seconds while the average path length remains similar to that obtained with a population size of 75. Thus, it can be inferred that an excessively large population size is not necessarily beneficial in improving the solution, and instead, it increases the computational power required.

4.3.2 Inertial Weight

The inertial weight is a parameter that controls the trade-off between the searching ability and convergence speed of the algorithm, it is a crucial parameter that balance between global and local search ability of the algorithm. The inertial weight determines the influence of the particle's previous velocity on its new velocity where a high value of the inertial weight causes the particles to move towards their current best position, allowing for faster convergence, but it also reduces the exploration ability of the MPSO algorithm. In the context of multi-robot path planning, the inertial weight can also have significant impact on the simulation result as it affects the searching process by the rate of convergence and exploration of the swarm. A high value of inertial weight allows the particles to move more freely and explore the search space which at the same time increase the chance in finding better solution. However, it should be noted that higher inertial weight may cause the particles to overshoot the optimal solution time and increase the number of iterations required to reach the optimum solution as well as the convergence time. On the other hand, a low value of inertial weight can lead to faster convergence but has higher risk in converging to local optimum and trap in local optimum.

4.3.3 Cognitive and Social Learning Factor

Cognitive and social learning factor are two other important parameters in PSO algorithm. These two parameters are crucial in determining the ability of the PSO algorithm in exploring the search space and converge toward the global optimum solution. The cognitive learning factor determine how much the particles learn from its personal best solution while the social learning factor determines how much the particles learn from the global best solution found by the particle swarm.

When high cognitive learning factor is used, the particles rely more on its own personal best solution to adjust its velocity and position. On the other hand, when higher social learning factor is used, the particles rely more on the global best solution found by the particles swarm to adjust its velocity and position. When the cognitive learning factor is dominating, the algorithm may tend to explore the search space more widely due to the fact that the position of the particles relies more on the personal best solution. When the social learning factor is more dominating, the algorithm may tend to converge toward a local optimum solution and this local optimum solution may not be the global optimum solution.

In order to test the effect on the cognitive and social learning factor on the simulation result, different combination of cognitive and social learning factors was used. In the first simulation, the cognitive learning factor is set to 2 while the social learning factor is set to 0 which means that the algorithm solely relies on the personal best solution found by each particle to determine the position for each particle. The second simulation was done by setting the cognitive learning factor to 0 while the social learning factor is set to 2 which means that the algorithm will be solely rely on the global best solution found by the particles swarm to determine the position for each particle. All the simulations were done by setting the inertial weight to be 0.4 as the minimum inertial value used in this project is 0.4 to observe the result obtained with different value of cognitive and social learning factors.

In the first simulation as shown in Figure 4.25 where the cognitive learning factor is set to 2 while the social learning factor is set to 0, it can be observed that the particles were first initialized within a predefined search space around the robot's starting position. In each iteration, the particles try to

update its position depending on the personal best solution found thus far by the particles. It can be observed that the position of the particles is not converging toward a global optimum solution, instead, the particle stays at their own initialized position due to the fact that they only rely on their own personal best solution.

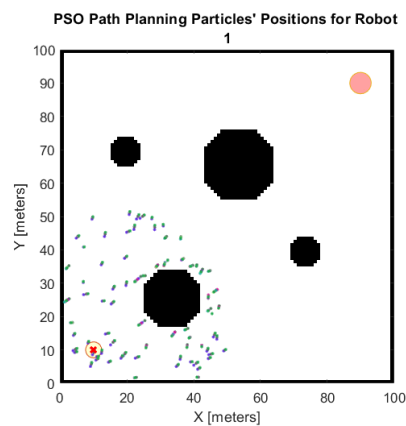


Figure 4.25: Result Obtained with different learning factor for 30 iterations
(Cognitive Learning Factor = 2, Social Learning Factor = 0)

In the second simulation as shown in Figure 4.26 where the cognitive learning factor is set to 0 while the social learning factor is set to 0, similarly, the particles were first initialized within a pre-defined search space around the robot's starting position and the position of the particles are updated iteratively to search for a global best solution within the search space. The particles with the same colour represent the particles in the same iteration. From the result, it can be observed that the particles can converge toward a solution by solely relying on the global best solution found by the particles swarm to update their position.

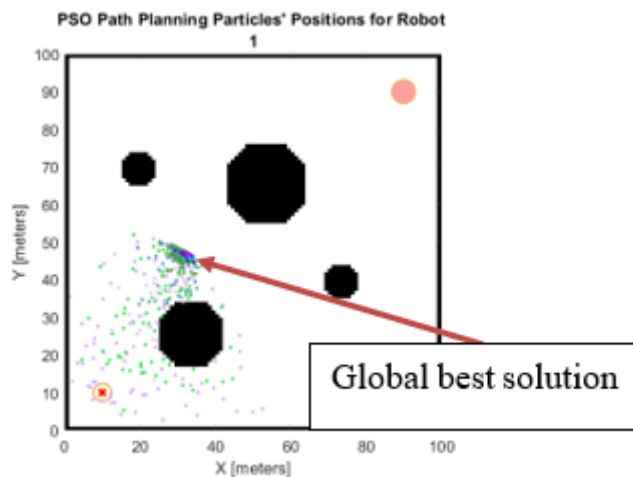


Figure 4.26: Result obtained for 30 iterations with cognitive learning factor = 0 and social learning factor = 2)

Therefore, it is suggested in Das, Behera and Panigrahi (2016) to dynamically update the cognitive and social learning factor in order to balance the exploration and exploitation ability of the algorithm.

4.4 Summary

The performance for both PSO and MPSO algorithms are evaluated in terms on average path length and execution time in three different scenarios. The biggest difference between MPSO and PSO algorithm is that MPSO algorithm incorporates a new path planning scheme, which PSO algorithm lacks. From the result obtained, it is indicated that the MPSO algorithm outperforms the PSO algorithm in all three scenarios, as it generates shorter average path length and execution time. Besides that, the impact of PSO parameters on the simulation results are examined. It reveals that MPSO algorithm has the best performance in terms of shorter path length and execution time with population size of 75. Additionally, it has been suggested that updating the inertial weight and learning factors dynamically can help maintain a balance between exploration and exploitation capability of the algorithm.

CHAPTER 5

CONCLUSIONS AND RECOMMENDATIONS

5.1 Conclusions

This project aimed to address the challenge of multi-robot path planning in cluttered environments, particularly in search and rescue missions. To achieve this, a modified version of the classical PSO algorithm, called MPSO algorithm, was proposed. Unlike the classical PSO algorithm which determine each of the robot's waypoint in each iteration of the algorithm, the MPSO algorithm incorporates a new path planning scheme that determines one robot's waypoint by running the entire algorithm once. In addition, a combination of global and local path planning approaches was employed to address the multi-robot path planning problem in dynamic environments. The PSO algorithm served as the global path planner to determine the complete path for each robot from respective starting position to respective goal position, while the Obstacle Avoidance Algorithm served as the local path planner to avoid collisions with dynamic obstacles during robot navigation.

Simulation was conducted using both PSO and MPSO algorithms in three scenarios for five times each, and the results showed that the MPSO algorithm outperformed the PSO algorithm in terms of average path length and execution time in all three scenarios. Moreover, the impact of the PSO parameters on the simulation results was also evaluated, and it was found that dynamically updating the inertial weight and learning factors could help maintain a balance between the exploration and exploitation capability of the algorithm. Furthermore, it was also found that the MPSO algorithm able to achieve the best performance with a population size of 75. Overall, the findings of this project suggest that the MPSO algorithm can generate better paths than the classical PSO algorithm in terms of average path length and execution time, and the proposed solution can effectively address the multi-robot path planning problem in dynamic environments.

5.2 Recommendations for Future Work

This project can be improved in many ways due to the project's limitation. It is important to note that the algorithm overlooks the smoothness aspect of the path generated. Consequently, the resultant path may exhibit abrupt directional changes, making it less smooth and challenging for the robot to navigate. Robot halting and rotation upon any change in direction is necessary due to lack of path smoothness, leading to increased travel time and higher energy consumption. There are a few options to enhance the smoothness of the path, such as adding a penalty function in the fitness function that penalizes sudden direction change when identifying the next waypoint or utilizing path smoothing techniques like spline interpolation or other smoothing algorithm after generating the path with MPSO (Das, Behera and Panigrahi, 2016c; Xu, Song and Cao, 2021).

Besides that, optimization algorithms, like the PSO algorithm, often encounter the issue of local optimum trapping. This happens when the algorithm converges to a solution that appears optimal within a particular region of the search space but is far from the true global optimum. The local optimum problem frequently arises when a complex-shaped obstacle is present in the environment, and the robot may become stuck at the obstacle and unable to escape the local optimum solution. Unfortunately, this project does not offer a solution to tackle the local optimum trapping problem, and the robot may become trapped in a more complex environment.

Moreover, the algorithm's performance in a physical environment remains untested, as it has only been evaluated in simulations. Thus, the performance of the algorithm can be further evaluated in the physical environment with Robotics Operating System (ROS). Furthermore, it is important to note that an efficient multi-robot system needs to be considered multiple aspects such as path planning, formation control, task allocation and so on. In this project, only the path planning problem is considered due to time constraint.

Therefore, future work could involve introducing path smoothing technique in the path planning algorithm, solving the local optimum trapping

problem, and extending the algorithm to physical experiments using Robotics Operating System (ROS) to validate its performance in a real-world scenario.

REFERENCES

Abdalla, T.Y., Abed, A.A. and Ahmed, A.A., 2017. ‘Mobile robot navigation using PSO-optimized fuzzy artificial potential field with fuzzy control’, *Journal of Intelligent & Fuzzy Systems*, 32(6), pp. 3893–3908. Available at: <https://doi.org/10.3233/IFS-162205>.

de Almeida, J.P.L.S. *et al.*, 2020. ‘A Global/Local Path Planner for Multi-Robot Systems with Uncertain Robot Localization’, *Journal of Intelligent & Robotic Systems*, 100(1), pp. 311–333. Available at: <https://doi.org/10.1007/s10846-020-01196-y>.

altigator (no date) *Drones for search & rescue missions*.

Atiyah, A.N., Adzhar, N. and Jaini, N.I., 2021. ‘An overview: on path planning optimization criteria and mobile robot navigation’, *Journal of Physics: Conference Series*, 1988(1), p. 012036. Available at: <https://doi.org/10.1088/1742-6596/1988/1/012036>.

Awan-Ur-Rahman, 2020. *Introduction to Ant colony optimization(ACO)*.

Ayari, A. and Bouamama, S., 2017. ‘A new multi-robot path planning algorithm: Dynamic distributed particle swarm optimization’, in *2017 IEEE International Conference on Real-time Computing and Robotics (RCAR)*. IEEE, pp. 437–442. Available at: <https://doi.org/10.1109/RCAR.2017.8311901>.

Becky Oskin, 2022. *Japan earthquake & tsunami of 2011: Facts and information*.

Biswas, S., Anavatti, S.G. and Garratt, M.A., 2017. ‘Obstacle Avoidance for Multi-agent Path Planning Based on Vectorized Particle Swarm Optimization’, in, pp. 61–74. Available at: https://doi.org/10.1007/978-3-319-49049-6_5.

Borenstein, J. and Koren, Y., 1989. ‘Real-time obstacle avoidance for fast mobile robots’, *IEEE Transactions on Systems, Man, and Cybernetics*, 19(5), pp. 1179–1187. Available at: <https://doi.org/10.1109/21.44033>.

Britannica, T.E. of E., 2022. *Indian Ocean tsunami of 2004*.

Bruno Siciliano and Oussama Khatib, 2007. *Springer Handbook of Robotics*. Berlin, Heidelberg: Springer-Verlag.

Chitikena, H., Sanfilippo, F. and Ma, S., 2023. ‘Robotics in Search and Rescue (SAR) Operations: An Ethical and Design Perspective Framework for Response Phase’, *Applied Sciences*, 13(3), p. 1800. Available at: <https://doi.org/10.3390/app13031800>.

Connell, D. and Manh La, H., 2018. ‘Extended rapidly exploring random tree-based dynamic path planning and replanning for mobile robots’, *International Journal of Advanced Robotic Systems*, 15(3), p. 172988141877387. Available at: <https://doi.org/10.1177/1729881418773874>.

Das, P.K. *et al.*, 2016. ‘An improved particle swarm optimization for multi-robot path planning’, in *2016 International Conference on Innovation and Challenges in Cyber Security (ICICCS-INBUSH)*. IEEE, pp. 97–106. Available at: <https://doi.org/10.1109/ICICCS.2016.7542324>.

Das, P.K., Behera, H.S. and Panigrahi, B.K., 2016. ‘A hybridization of an improved particle swarm optimization and gravitational search algorithm for multi-robot path planning’, *Swarm and Evolutionary Computation*, 28, pp. 14–28. Available at: <https://doi.org/10.1016/j.swevo.2015.10.011>.

Das, P.K. and Jena, P.K., 2020. ‘Multi-robot path planning using improved particle swarm optimization algorithm through novel evolutionary operators’, *Applied Soft Computing*, 92, p. 106312. Available at: <https://doi.org/10.1016/j.asoc.2020.106312>.

Debnath, S.K. *et al.*, 2021. ‘Different Cell Decomposition Path Planning Methods for Unmanned Air Vehicles-A Review’, in, pp. 99–111. Available at: https://doi.org/10.1007/978-981-15-5281-6_8.

Doroftci, D., Matos, A. and de Cubber, G., 2014. ‘Designing Search and Rescue Robots towards Realistic User Requirements’, *Applied Mechanics and*

Materials, 658, pp. 612–617. Available at: <https://doi.org/10.4028/www.scientific.net/AMM.658.612>.

Drew, D.S., 2021. ‘Multi-Agent Systems for Search and Rescue Applications’, *Current Robotics Reports*, 2(2), pp. 189–200. Available at: <https://doi.org/10.1007/s43154-021-00048-3>.

Eberhart, R. and Kennedy, J., n.d. ‘A new optimizer using particle swarm theory’, in *MHS’95. Proceedings of the Sixth International Symposium on Micro Machine and Human Science*. IEEE, pp. 39–43. Available at: <https://doi.org/10.1109/MHS.1995.494215>.

Gautam, A. and Mohan, S., 2012. ‘A review of research in multi-robot systems’, in *2012 IEEE 7th International Conference on Industrial and Information Systems (ICIIS)*. IEEE, pp. 1–5. Available at: <https://doi.org/10.1109/ICIInfS.2012.6304778>.

Ge, Q. *et al.*, 2021. ‘Improved Bidirectional RRT $\langle \text{M1} \rangle$ Path Planning Method for Smart Vehicle’, *Mathematical Problems in Engineering*, 2021, pp. 1–14. Available at: <https://doi.org/10.1155/2021/6669728>.

Han, S., Zhou, X. and Chen, C., 2016. ‘Path planning for multi-robot systems using PSO and Critical Path Schedule Method’, in *2016 IEEE 13th International Conference on Networking, Sensing, and Control (ICNSC)*. IEEE, pp. 1–6. Available at: <https://doi.org/10.1109/ICNSC.2016.7478999>.

Iran Macedo, 2018. *Implementing the Particle Swarm Optimization (PSO) Algorithm in Python*.

Jing Ren, McIsaac, K.A. and Patel, R.V., 2006. ‘Modified Newton’s method applied to potential field-based navigation for mobile robots’, *IEEE Transactions on Robotics*, 22(2), pp. 384–391. Available at: <https://doi.org/10.1109/TRO.2006.870668>.

Khatib, O., 1986. ‘Real-time obstacle avoidance for manipulators and mobile robots’, in *Proceedings. 1985 IEEE International Conference on Robotics and*

Automation. Institute of Electrical and Electronics Engineers, pp. 500–505. Available at: <https://doi.org/10.1109/ROBOT.1985.1087247>.

Kim, J.-O. and Khosla, P.K., 1992. ‘Real-time obstacle avoidance using harmonic potential functions’, *IEEE Transactions on Robotics and Automation*, 8(3), pp. 338–349. Available at: <https://doi.org/10.1109/70.143352>.

Koren, Y. and Borenstein, J., n.d. ‘Potential field methods and their inherent limitations for mobile robot navigation’, in *Proceedings. 1991 IEEE International Conference on Robotics and Automation*. IEEE Comput. Soc. Press, pp. 1398–1404. Available at: <https://doi.org/10.1109/ROBOT.1991.131810>.

Koubaa, A. *et al.*, 2018. ‘Introduction to Mobile Robot Path Planning’, in, pp. 3–12. Available at: https://doi.org/10.1007/978-3-319-77042-0_1.

Lamini, C., Fathi, Y. and Benhlima, S., 2017. ‘H-MAS architecture and reinforcement learning method for autonomous robot path planning’, in *2017 Intelligent Systems and Computer Vision (ISCV)*. IEEE, pp. 1–7. Available at: <https://doi.org/10.1109/ISACV.2017.8054978>.

Lee, H., Lee, D. and Shim, D.H., 2017. ‘Receding Horizon-based RRT* Algorithm for a UAV Real-time Path Planner’, in *AIAA Information Systems-AIAA Infotech @ Aerospace*. Reston, Virginia: American Institute of Aeronautics and Astronautics. Available at: <https://doi.org/10.2514/6.2017-0676>.

Li, X. *et al.*, 2020. ‘An Improved Method of Particle Swarm Optimization for Path Planning of Mobile Robot’, *Journal of Control Science and Engineering*, 2020, pp. 1–12. Available at: <https://doi.org/10.1155/2020/3857894>.

Liu, S., Mao, L. and Yu, J., 2006. ‘Path Planning Based on Ant Colony Algorithm and Distributed Local Navigation for Multi-Robot Systems’, in *2006 International Conference on Mechatronics and Automation*. IEEE, pp. 1733–1738. Available at: <https://doi.org/10.1109/ICMA.2006.257476>.

Mac, T.T. *et al.*, 2016. 'Heuristic approaches in robot path planning: A survey', *Robotics and Autonomous Systems*, 86, pp. 13–28. Available at: <https://doi.org/10.1016/j.robot.2016.08.001>.

Maryam Yarmohamadi and Hossein Erfani, 2011. 'Improvement of Robot Path Planning Using Particle Swarm Optimization in Dynamic Environments with Mobile Obstacles and Target', *Advanced Studies in Biology*, 3(1), pp. 43–53.

Matoui, F. *et al.*, 2017. 'Path planning of a group of robots with potential field approach: decentralized architecture', *IFAC-PapersOnLine*, 50(1), pp. 11473–11478. Available at: <https://doi.org/10.1016/j.ifacol.2017.08.1822>.

Melchior, N.A. and Simmons, R., 2007. 'Particle RRT for Path Planning with Uncertainty', in *Proceedings 2007 IEEE International Conference on Robotics and Automation*. IEEE, pp. 1617–1624. Available at: <https://doi.org/10.1109/ROBOT.2007.363555>.

Patle, B.K. *et al.*, 2019. 'A review: On path planning strategies for navigation of mobile robot', *Defence Technology*, 15(4), pp. 582–606. Available at: <https://doi.org/10.1016/j.dt.2019.04.011>.

Prasad, A.S. and Francescutti, L.H., 2017. 'Natural Disasters', in *International Encyclopedia of Public Health*. Elsevier, pp. 215–222. Available at: <https://doi.org/10.1016/B978-0-12-803678-5.00519-1>.

Queralta, J.P. *et al.*, 2020. 'Collaborative Multi-Robot Search and Rescue: Planning, Coordination, Perception, and Active Vision', *IEEE Access*, 8, pp. 191617–191643. Available at: <https://doi.org/10.1109/ACCESS.2020.3030190>.

Shi, Y. and Eberhart, R., n.d. 'A modified particle swarm optimizer', in *1998 IEEE International Conference on Evolutionary Computation Proceedings. IEEE World Congress on Computational Intelligence (Cat. No.98TH8360)*. IEEE, pp. 69–73. Available at: <https://doi.org/10.1109/ICEC.1998.699146>.

Siobhan Grayson, 2014. 'Search & Rescue using Multi-Robot Systems'.

Sun, J. *et al.*, 2019. ‘Smart Obstacle Avoidance Using a Danger Index for a Dynamic Environment’, *Applied Sciences*, 9(8), p. 1589. Available at: <https://doi.org/10.3390/app9081589>.

TAN, G.-Z., 2007. ‘Ant Colony System Algorithm for Real-Time Globally Optimal Path Planning of Mobile Robots’, *ACTA AUTOMATICA SINICA*, 33(3), p. 0279. Available at: <https://doi.org/10.1360/aas-007-0279>.

Tang, B. *et al.*, 2020. ‘Multi-robot path planning using an improved self-adaptive particle swarm optimization’, *International Journal of Advanced Robotic Systems*, 17(5), p. 172988142093615. Available at: <https://doi.org/10.1177/1729881420936154>.

Tim Chinenov, 2019. *Robotic Path Planning: RRT and RRT**.

Tzafestas, S.G., 2014. ‘Mobile Robot Path, Motion, and Task Planning’, in *Introduction to Mobile Robot Control*. Elsevier, pp. 429–478. Available at: <https://doi.org/10.1016/B978-0-12-417049-0.00011-0>.

United Nation, 2021. *Climate and weather-related disasters surge five-fold over 50 years, but early warnings save lives - WMO report*.

United Nation, 2022. *UN marks anniversary of devastating 2010 Haiti earthquake*.

Wu, Z., Su, W. and Li, J., 2019. ‘Multi-robot path planning based on improved artificial potential field and B-spline curve optimization’, in *2019 Chinese Control Conference (CCC)*. IEEE, pp. 4691–4696. Available at: <https://doi.org/10.23919/ChiCC.2019.8865232>.

Wu, Zhenping *et al.*, 2021. ‘Fast-RRT: A RRT-Based Optimal Path Finding Method’, *Applied Sciences*, 11(24), p. 11777. Available at: <https://doi.org/10.3390/app112411777>.

Xu, L., Song, B. and Cao, M., 2021. ‘A new approach to optimal smooth path planning of mobile robots with continuous-curvature constraint’, *Systems*

Science & Control Engineering, 9(1), pp. 138–149. Available at: <https://doi.org/10.1080/21642583.2021.1880985>.

Yang Li *et al.*, 2013. ‘Multi-robot path planning based on the developed RRT* algorithm’, in *Proceedings of the 32nd Chinese Control Conference*, pp. 7049–7053.

Zhang, H., Lin, W. and Chen, A., 2018. ‘Path Planning for the Mobile Robot: A Review’, *Symmetry*, 10(10), p. 450. Available at: <https://doi.org/10.3390/sym10100450>.

Zhu, Q., Yan, Y. and Xing, Z., 2006. ‘Robot Path Planning Based on Artificial Potential Field Approach with Simulated Annealing’, in *Sixth International Conference on Intelligent Systems Design and Applications*. IEEE, pp. 622–627. Available at: <https://doi.org/10.1109/ISDA.2006.253908>.

Zurich, 2022. *Is climate change making natural hazards worse?*

APPENDICES

Appendix A: Pseudocode

PseudocodeA-1: Pseudocode for Path Planning using MPSO.

Algorithm 1: Pseudocode for Path Planning using MPSO

Input: $(x_j^{start}, y_j^{start}), (x_j^{target}, y_j^{target})$ are the initial position and target position for n number of robots where $1 \leq j \leq n$

Output: Optimum trajectory path for each robot connecting each waypoint, wp_j from (x_j^{curr}, y_j^{curr}) to $(x_j^{target}, y_j^{target})$

Initialize PSO parameters: SwarmSize, $iter_{total}$, DimensionNo, c_{1max} , c_{1min} , c_{2max} , c_{2min} , ω_{max} , ω_{min} , V_{max} , V_{min}

For $j = 1$: no_of_robots

| $(x_j^{curr}, y_j^{curr}) = (x_j^{start}, y_j^{start}); wp_j = [(x_j^{curr}, y_j^{curr})];$

End for

For each robot $j = 1$: RobotNo

| **While** $(x_j^{curr}, y_j^{curr}) - (x_j^{target}, y_j^{target}) > tolerance$ **do**

| **For** $i = 1$: SwarmSize

| Initialize i^{th} particle with random position and random velocity within a predefined search space;

| **End For**

| **For** $iter_{ctr} = 1$: $iter_{total}$

| **For** $n = 1$: DimensionNo

| **For** $i = 1$: SwarmSize

| Evaluate fitness value for i^{th} particle;

| **If** $(fit_i > fit(P_{ibest}))$

| $fit(P_{ibest}) = fit_i;$

| Current position of i^{th} particle = $P_{ibest};$

| **End If**

| **If** $(fit(P_{ibest}) > fit(G_{best}))$

| $fit(G_{best}) = fit(P_{ibest});$

| $G_{best} = P_{ibest};$

| **End If**

| **End For**

| **End For**

| Update the velocity and position of i^{th} particle;

| Update ω , c_1 and c_2 using equations;

| **If** G_{best} is not updated for three times

| Break;

| **End If**

| **End For**

| $(x_j^{curr}, y_j^{curr}) = G_{best};$

| $wp_j = [waypoint_j; (x_j^{curr}, y_j^{curr})];$

| **End While**

End For

PseudocodeA-2: Pseudocode for Obstacle Avoidance Algorithm.

Algorithm 2: Pseudocode for Obstacle Avoidance Algorithm

Input: Robot's position, (Robots(j).pos), position of dynamic obstacles, sensing range

Output: Robot's position, (Robots(j).pos)

For j = 1: RobotNo

While Robots(j).reached == true

 Reset sensors' value to 0, Robots(j).sensor_value(k) = 0;

For m = 1: number of dynamic obstacles

If distance between obstacle(m) and Robots(j).pos < sensing_range

 theta = angle between Robots(j).pos and the position of dynamic obstacle;

For k = 1: number of sensors

If theta within Robots(j).sensing_range(k)

 Robots(j).sensor_value(k) = 1;

End If

End For

End If

End For

For k = 1: SensorNo

If Robots(j).sensor_value(k) = 1

 Sensor_pos(k).x = x-coordinate of possible next position based on direction of kth sensor;

 Sensor_pos(k).y = y-coordinate of possible next position based on direction of kth sensor;

 Distance(k) = distance between sensor_pos(k) and robot's next waypoint;

Else

 Distance(k) = inf;

End If

End For

 index = index of sensor with minimum Distance(k);

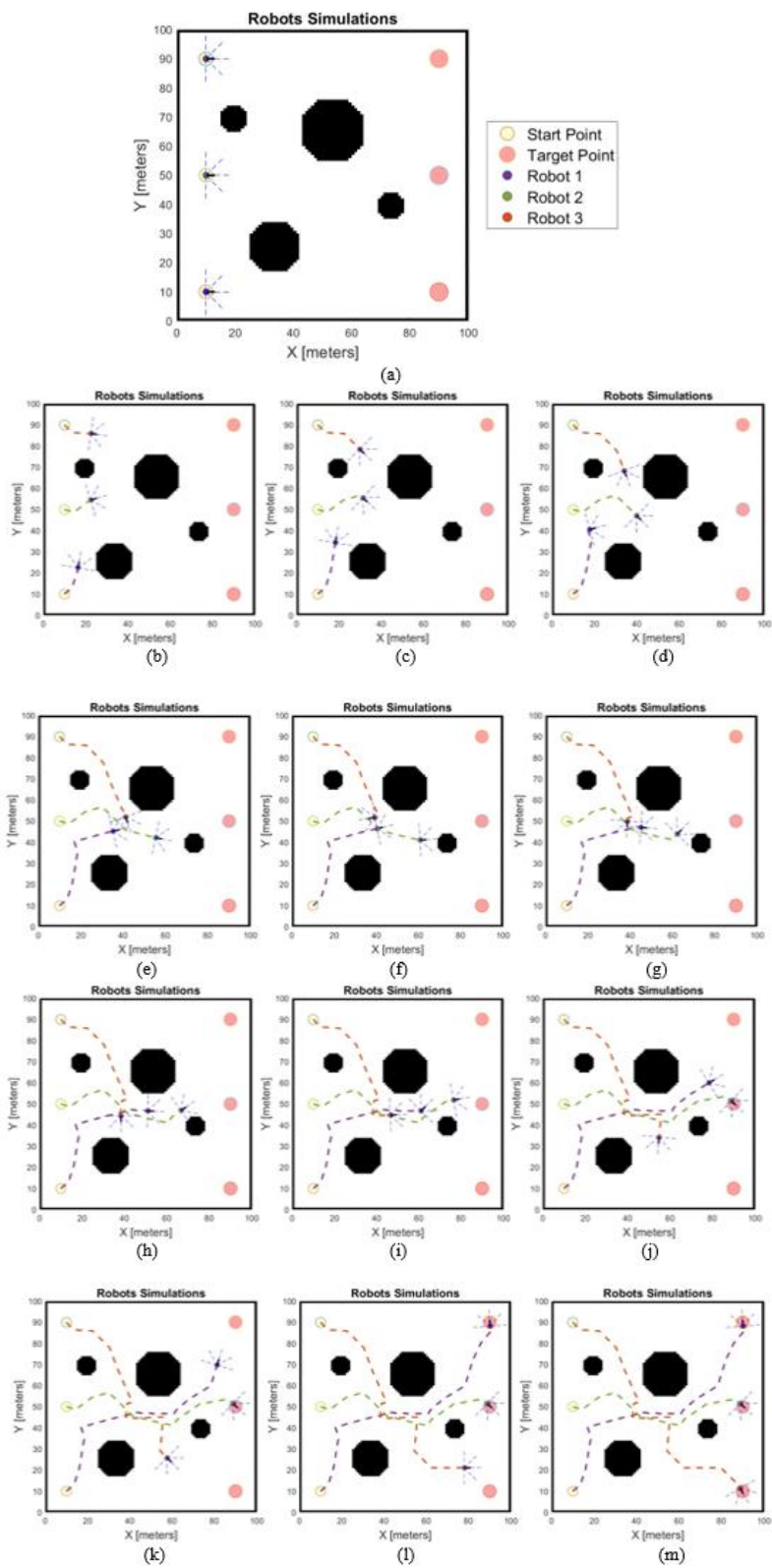
 Robots(j).pos.x = sensor_pos(index).x;

 Robots(j).pos.y = sensor_pos(index).y;

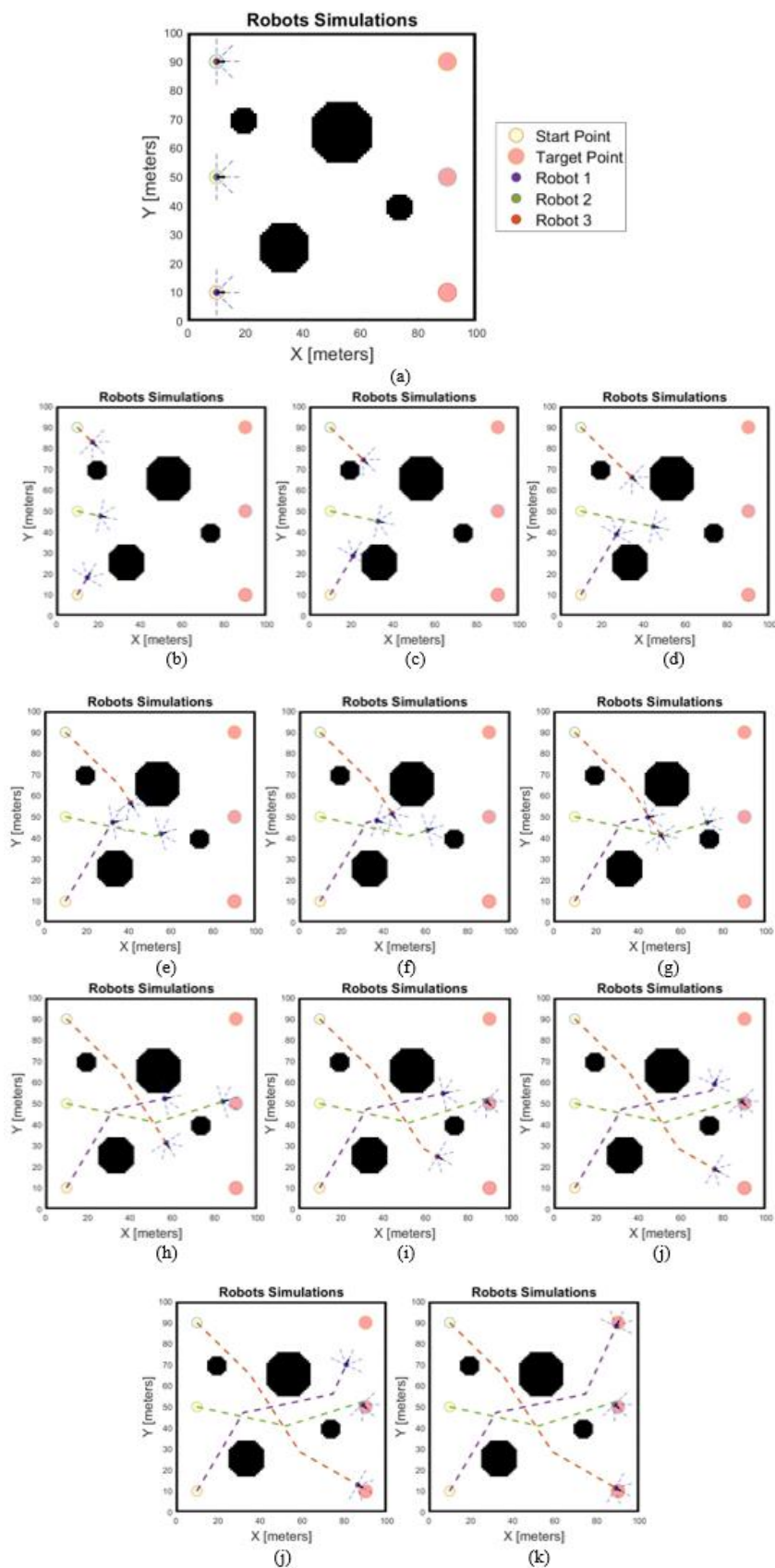
End While

End For

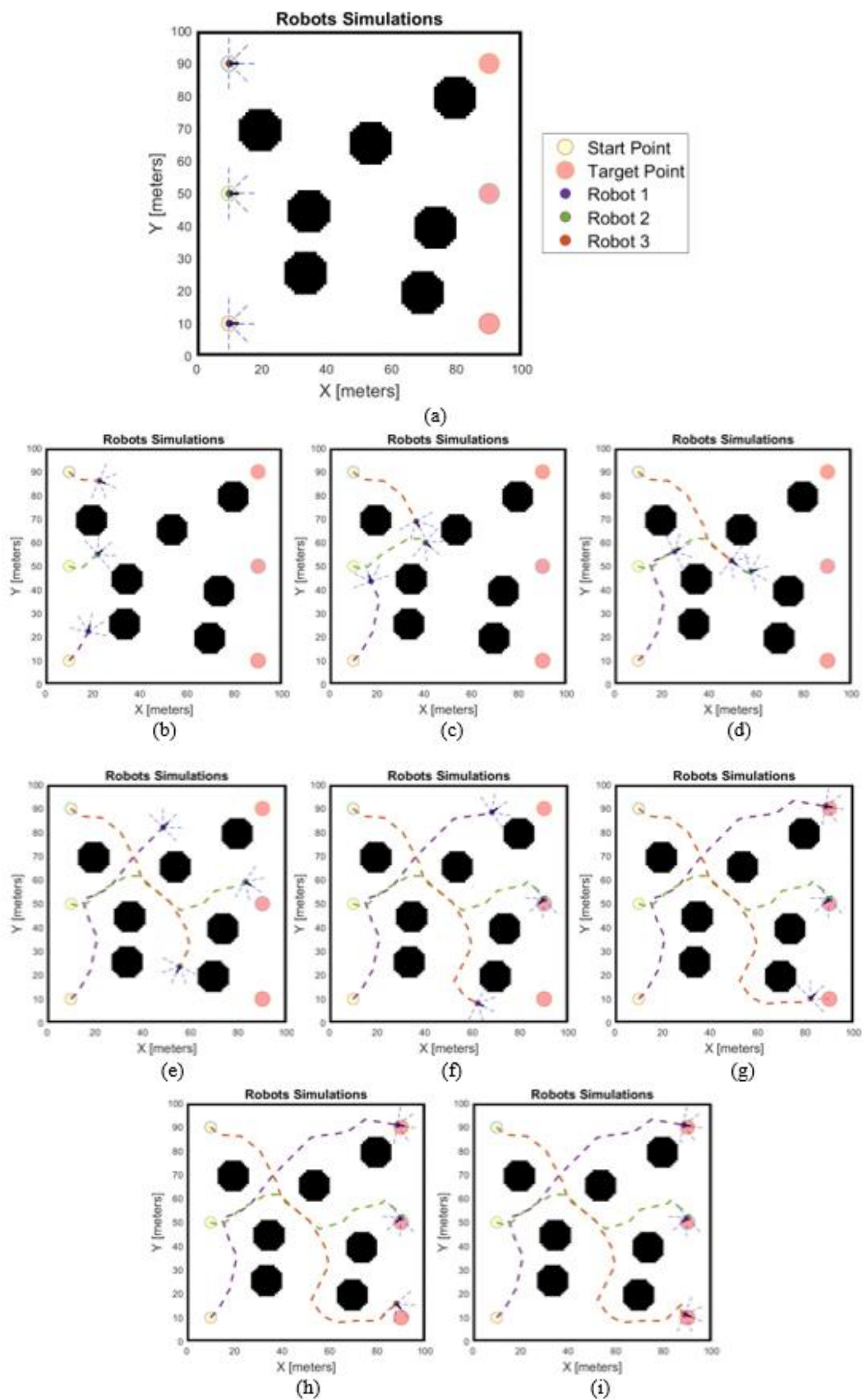
Appendix B: Simulations



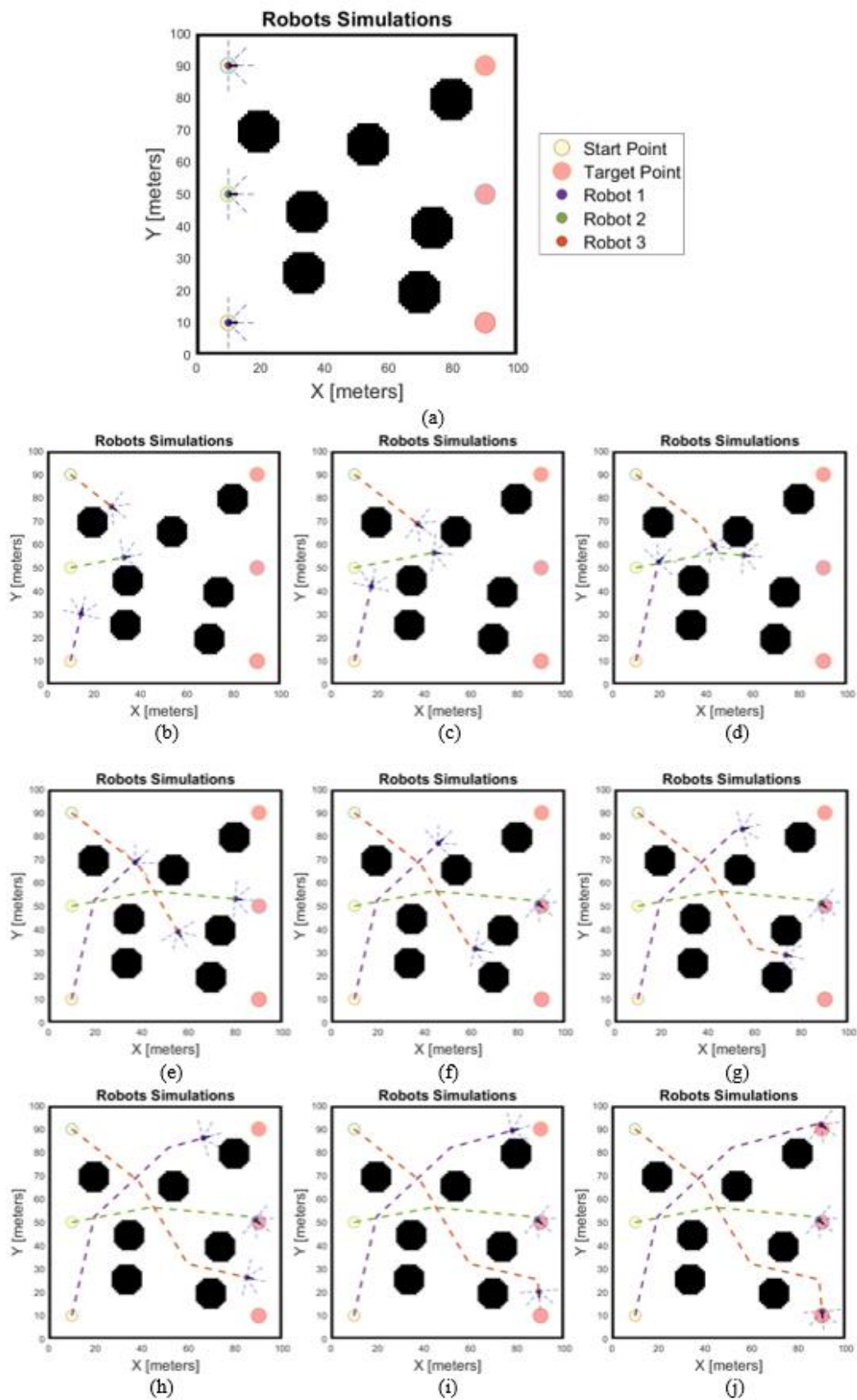
SimulationB-1: PSO Simulation in Scenario 1 From Simulation Step 1 in (a) to Step 10 in (m)

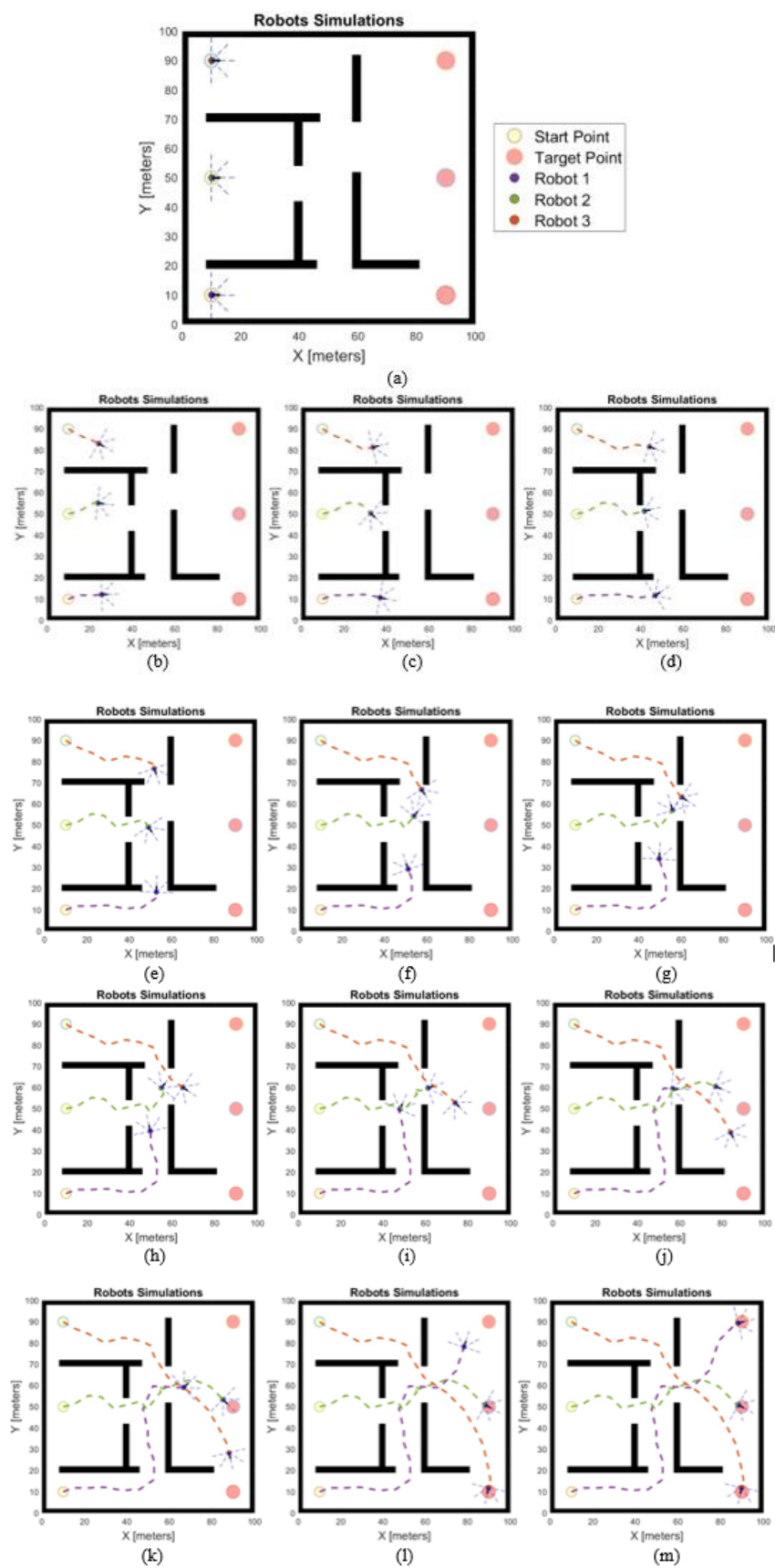


SimulationB-2: MPSO Simulation in Scenario 1 From Simulation Step 1 in (a) to Step 12 in (k)

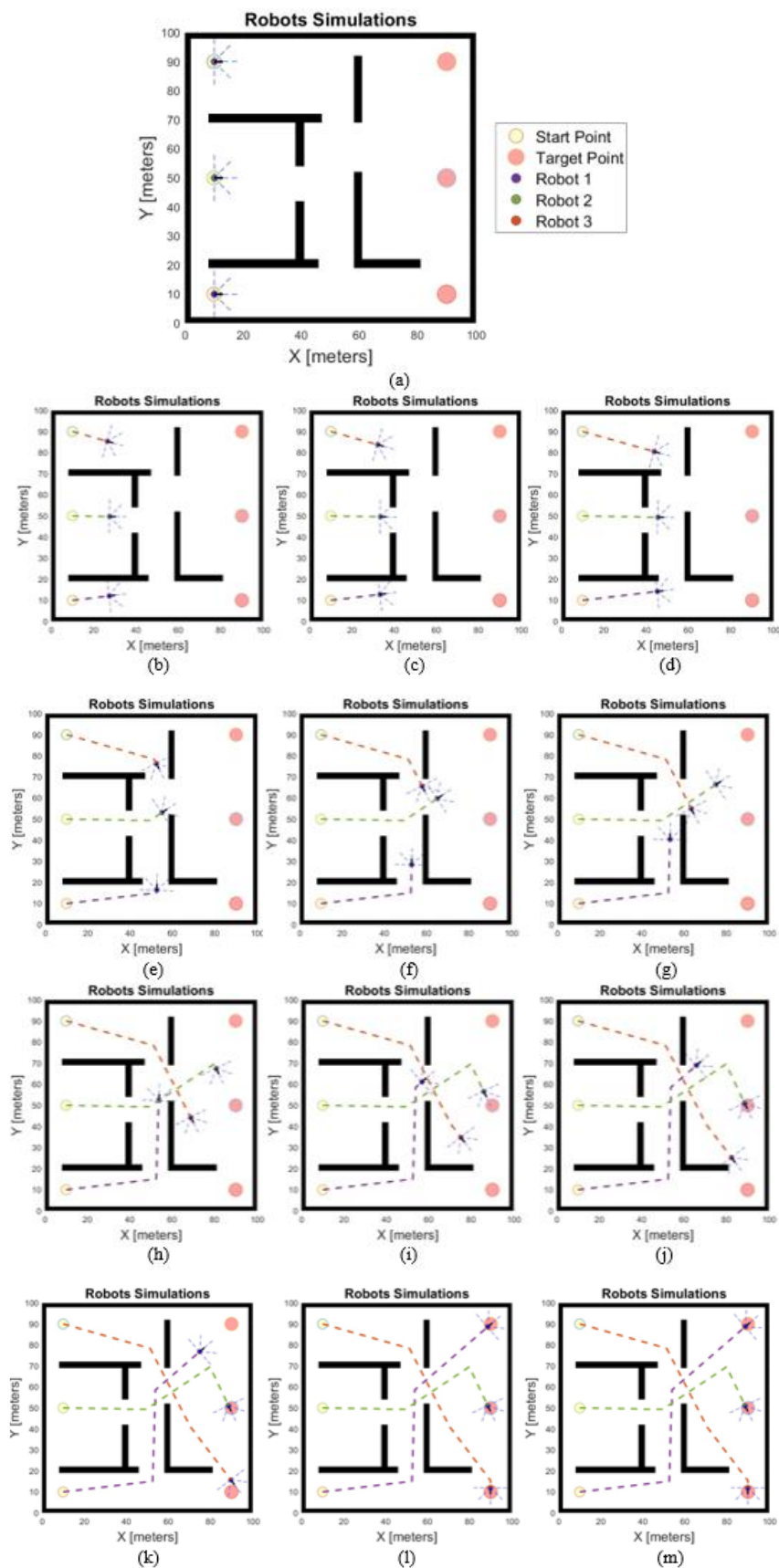


SimulationB-3: PSO Simulation in Scenario 2 From Simulation Step 1 in (a) to Step 9 in (i)



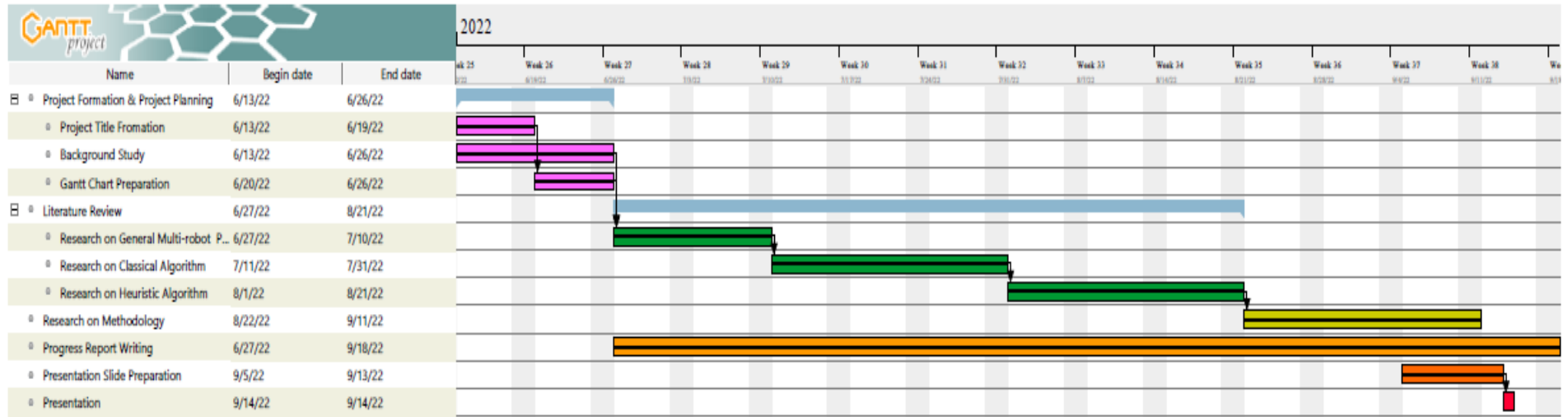


SimulationB-5: PSO Simulation in Scenario 3 From Simulation Step 1 in (a) to Step 13 in (m)



SimulationB-6: MPSO Simulation in Scenario 3 From Simulation Step 1 in (a) to Step 13 in (m)

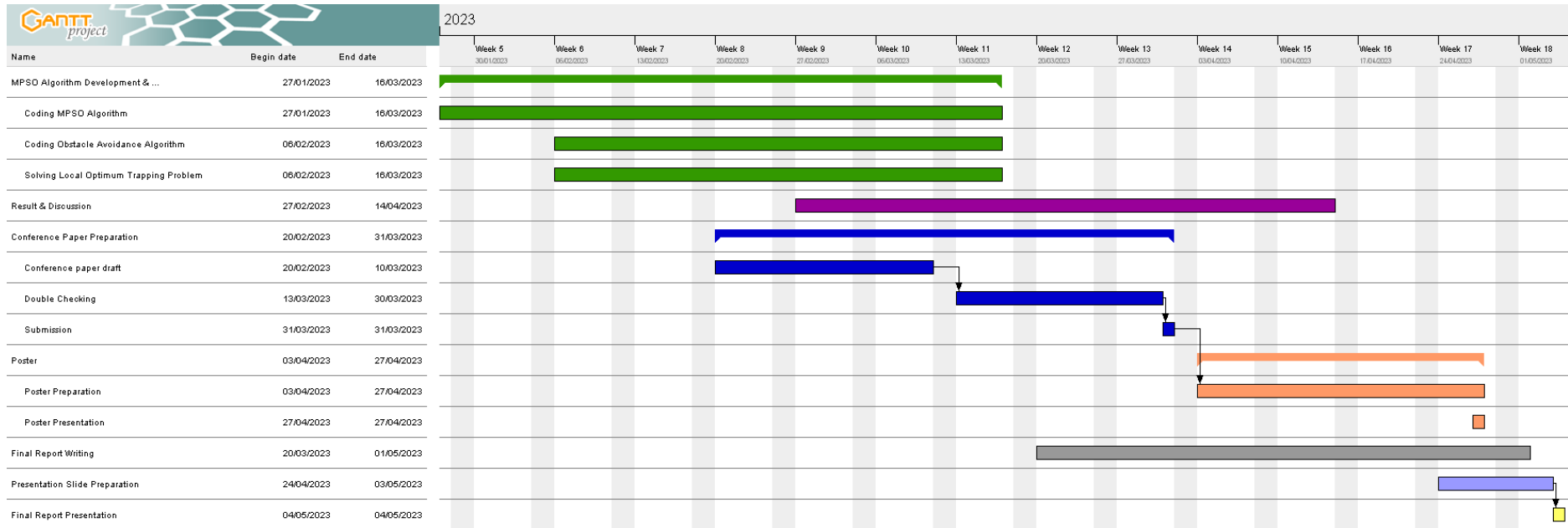
Appendix C: Gantt Chart



GanttChartC-1: Gantt Chart Using GanttProject (Part I)

Name	Begin date	End date
▼ Project Formation & Project Plan...	13/06/2022	26/06/2022
Project Title Fromation	13/06/2022	19/06/2022
Background Study	13/06/2022	26/06/2022
Gantt Chart Preparation	20/06/2022	26/06/2022
▼ Literature Review	27/06/2022	21/08/2022
Research on General Multi-ro...	27/06/2022	10/07/2022
Research on Classical Algorithm	11/07/2022	31/07/2022
Research on Heuristic Algorithm	01/08/2022	21/08/2022
Research on Methodology	22/08/2022	11/09/2022
Progress Report Writing	27/06/2022	18/09/2022
Presentation Slide Preparation	05/09/2022	13/09/2022
Presentation	14/09/2022	14/09/2022

GanttChartC-2: Description of Gantt Chart Using GanttProject (Part I)



GanttChartC-3: Gantt Chart Using GanttProject (Part II)

Name	Begin date	End date
▼ MPSO Algorithm Development ...	27/01/2023	16/03/2023
Coding MPSO Algorithm	27/01/2023	16/03/2023
Coding Obstacle Avoidance A...	06/02/2023	16/03/2023
Solving Local Optimum Trapp...	06/02/2023	16/03/2023
Result & Discussion	27/02/2023	14/04/2023
▼ Conference Paper Preparation	20/02/2023	31/03/2023
Conference paper draft	20/02/2023	10/03/2023
Double Checking	13/03/2023	30/03/2023
Submission	31/03/2023	31/03/2023
▼ Poster	03/04/2023	27/04/2023
Poster Preparation	03/04/2023	27/04/2023
Poster Presentation	27/04/2023	27/04/2023
Final Report Writing	20/03/2023	01/05/2023
Presentation Slide Preparation	24/04/2023	03/05/2023
Final Report Presentation	04/05/2023	04/05/2023

GanttChartC-4: Description of Gantt Chart Using GanttProject (Part II)