



REPUBLIC OF TÜRKİYE

ALTINBAŞ UNIVERSITY

Institute of Graduate Studies

Electrical and Computer Engineering

**MULTI-AGENT PATH PLANNING FOR A
MOBILE ROBOT IN AN UNKNOWN DYNAMIC
ENVIRONMENT**

Mustafa Mohammed Hasan ALHASSOW

Doctor of Philosophy

Supervisor

Assoc. Prof. Dr. Oğuz ATA

İstanbul, 2024

MULTI-AGENT PATH PLANNING FOR A MOBILE ROBOT IN AN UNKNOWN DYNAMIC ENVIRONMENT

Mustafa Mohammed Hasan ALHASSOW

Electrical and Computer Engineering

Ph. D. Thesis

ALTINBAŞ UNIVERSITY

2024

The dissertation titled MULTI-AGENT PATH PLANNING FOR A MOBILE ROBOT IN AN UNKNOWN DYNAMIC ENVIRONMENT prepared by MUSTAFA ALHASSOW and submitted on has been **accepted unanimously** for the degree of PhD in Electrical and Computer Engineering.

Assoc.Prof.Dr. Doęu Çaędaş ATILLA

Co-Supervisor

Assoc. Prof. Dr. Oęuz ATA

Supervisor

Thesis Defense Committee Members:

Doę. Dr. Oęuz ATA

Department of Software
Engineering

Altinbas University

Doę. Dr. Sefer KURNAZ

Department of Computer
Engineering,

Altinbas University

Prof. Dr. Osman NURI UęAN

Department of Electrical and
Electronics engineering,

Altinbas University

Doę. Dr. Aytuę BOYACI

Department of Computer
Engineering,

National Defense University

Dr. Öęr. Üyesi Timur İNAN

Department of Computer
engineering,

Marmara University

I hereby declare that this thesis/dissertation meets all format and submission requirements of a PhD dissertation.

I hereby declare that all information/data presented in this graduation project has been obtained in full accordance with academic rules and ethical conduct. I also declare all unoriginal materials and conclusions have been cited in the text and all references mentioned in the Reference List have been cited in the text, and vice versa as required by the abovementioned rules and conduct.

Mustafa Mohammed Hasan ALHASSOW

Signature

DEDICATION

I dedicate this letter to my loved family and friends, whose continued assistance has been priceless to me. I greatly appreciate my dear parents Mohammed Hasan and Tases Alzubaidi. Through his books, my father instilled in me an avid interest in the search for knowledge while my mother deprived herself of many things in our interests so that my brothers and I could study and learn.

I would like to thank my beloved wife, Duha Ahmed, for being by my side through life's pressures and stresses.

I would also like to extend my most sincere words of thanks to all my friends, the moral support from whom has been a constant source of encouragement right from the very beginning. I would also like to express my continued gratitude to Dr. Oğuz Ata and Dr. Dogu Cadas Atilla for their continued help. Their care and follow-up in the journey of research have been second to none.

PREFACE

Permit me to express my heartfelt gratitude for the good hospitality I enjoyed in Turkey and Altinbas University, who provided the wherewithal to put this thesis together. My thanks go to Dr. Oguz Ata and Dr. Dogu Cadas Atila, who have been engaging throughout continuous guidance on research and writing of the thesis.



ABSTRACT

MULTI-AGENT PATH PLANNING FOR A MOBILE ROBOT IN AN UNKNOWN DYNAMIC ENVIRONMENT

ALHASSOW, Mustafa Mohammed Hasan

PhD, Electrical and Computer Engineering, Altınbaş University,

Supervisor: Assoc. Prof. Dr. Oğuz ATA

Co-Supervisor: Assoc. Prof. Dr. Doğu Çağdaş ATILLA

Date: 12/2024

Pages: 123

This work addresses the problem of robot path planning for multi-agent systems. It involves the process of enabling agents to move around simultaneously without collisions. Each agent has its destination to reach in an environment shared by all agents. However, unexpected changes in either the map or the goal in dynamic and unpredictable environments could lead to problems in performances or complete failures to achieve one's goals. In consequence, priority inheritance also proved to be very important in such cases. Efficiency in path planning and/or adaptation to sudden changes is impressively determined by priority inheritance. Regarding these two challenges, this research introduces a novel Conflict-Based Search approach and presents a new hierarchical search mechanism optimally designed for multi-robot path planning. Naturally, the most important aim is to increase the flexibility and adaptiveness of an algorithm in various environments. It also tests its efficacy in three different scenarios: the first one considering the application of the algorithm in unknown stationary and mobile environments yields excellent results, averaging 2.3 seconds of time usage with efficient path length optimization. The second scenario was followed by complex scenes with sharp corners and unknown obstacles, for which the average arrival time in this scenario was 2.6 seconds by the algorithm, the path length remaining optimal. The final scenario was performed in a warehouse environment with fixed, mobile, and multi-target

obstacles; the algorithm was performing well, with a maximum time of 100.4 s under the most complex settings. Various pathfinding algorithms, such as PRM, RRT, A*, D*, Dijkstra, GA, ACO, PSO, VO, and QL, have their respective pros, cons, and limitations concerning the environment, complexity, and the number of agents in the MAPF domain were compared and demonstrate that the proposed CBS approach is highly effective, efficient, and versatile, hence suitable for a wide range of difficult environments.

Keywords: Mobile Robot; Multi-Agent; Path Finding; Avoidance Capability; Dynamic Environment; Omnidirectional Agent.



TABLE OF CONTENTS

	<u>Pages</u>
ABSTRACT	vii
LIST OF TABLES.....	xi
LIST OF FIGURES.....	xii
ABBREVIATIONS.....	xv
LIST OF SYMBOLS.....	xvii
1. INTRODUCTION	1
1.1 NAVIGATION SYSTEM	1
1.2 MOBILE ROBOT PATH PLANNING.....	3
1.3 MULTI-AGENT PATH PLANNING	7
1.4 STATIC ENVIRONMENT OVERVIEW.....	9
1.5 DYNAMIC ENVIRONMENTS OVERVIEW.....	11
1.6 CHALLENGES AMONG PATH PLANNING.....	14
1.7 MOTIVATION.....	18
1.8 OBJECTIVES.....	18
1.9 CONTRIBUTION	19
1.10 ORGANIZATION	20
2. RELATED WORKS.....	21
2.1 SEARCH STRUCTURE	21
2.2 GRAPH SEARCH STRUCTURE.....	23
2.2.1 Path Planning Methods Within Graph Search	25
2.3 OUR PUBLISHED PAPERS RELATED TO THE GRAPH SEARCH STRUCTURE:	30
2.3.1 Nonholonomic Path Planning for A Mobile Robot Based on Voronoi And Q- Learning Algorithm	30
2.3.2 Car-Like Robot Path Planning Based on Voronoi and Q-Learning Algorithms .	33

2.4 MULTI-AGENT PATHFINDING	37
2.5 COLLISION FREE PATH WITH PICK AND DELIVERY OPERATION	40
2.6 LITERATURE REVIEW	43
2.7 SUMMARY	45
3. METHODOLOGY	47
3.1 MULTI-AGENT PATHFINDING	47
3.2 COST METRICS	49
3.3 CONFLICT-BASED SEARCH (CBS)[1].....	50
3.4 GENERAL PATH METHOD	57
4. SIMULATION STRUCTURE.....	61
4.1 IMPLIMENTATION SETTINGS	61
4.2 EXPERIMENTS SCENARIOES	64
4.2.1 First Scenario	65
4.2.2 Second Scenario.....	73
4.2.3 Third Scenario in A Warehouse Environment.....	80
5. DISCUSSION.....	90
6. CONCLUSION	97
REFERENCES	98

LIST OF TABLES

	<u>Pages</u>
Table 2.1: Illustrate Related Surveys on The Avoidance Of Obstacles In A Different Area.....	45
Table 4.1: Result Description of The Proposed Approach.....	68
Table 4.2: Simulation Result Under Dynamic Movement.....	74
Table 4.3: Describe The New Goal Position That Have Been Occurred to The Mobile Robots.....	82
Table 4.4: The Results Describe the Warehouse Environment and The Change That Occurred Three Times in It.....	85
Table 4.5: Result and Comparison Between All the Changes That Occurred in The Environment.....	89
Table 5.1: The Key Advantage of The Related Method.....	94
Table 5.2: The Comparison of Modified CBS Result with Mentioned Related Works.....	96

LIST OF FIGURES

	<u>Pages</u>
Figure 1.1: Path Planning Overview Example...	3
Figure 1.2: Global and Local Path Planning Classification.....	7
Figure 1.3: MAPF Overview Example.....	8
Figure 1.4: Dynamic Path Planning Example.....	12
Figure 1.5: Obstacle Avoidance Challenges.....	16
Figure 1.6: Warehouse Path Planning Example of Pickup And Delivery Operation [75]...	17
Figure 2.1: Shows An Example of The Start State And Goal State Inside The Environments [76].....	22
Figure 2.2: Shows An Example of Shortest Path Length.....	25
Figure 2.3: RRT Algorithm Structure.....	27
Figure 2.4: RRT* Structure [39].....	28
Figure 2.5: PRM Algorithm Structure [42].....	29
Figure 2.6: Free Path Inside the Environment Planning.....	31
Figure 2.7: The Under-Test Environment.....	32
Figure 2.8: The Result Obtained by The Proposed Work.....	33
Figure 2.9: Voronoi Diagram Environment Result.....	35
Figure 2.10: Result Obtained by Proposed Method.....	36
Figure 2.11: Shows An Example of MAPF Inside The Environment.....	40
Figure 3.1: MAPF Map Overview.....	48
Figure 3.2: Conflict Resolution In MCBS.....	52

Figure 3.3: Example Of Tree Node of The Multi-Agent Path.....	55
Figure 3.4: Efficient Path Planning for Robots.....	59
Figure 3.5: Shows The Work Structure.....	60
Figure 4.1: Environment Setting.....	61
Figure 4.2: Each Agent Setting Goals.....	62
Figure 4.3: Target Assignment Map.....	63
Figure 4.4: Shows Example of Target Assignment Within The Environment [45].....	64
Figure 4.5: Experiment Scenario Process Map.....	65
Figure 4.6: First Scenario Example.....	66
Figure 4.7: Example of the First Scenario as A Simple Environment.....	69
Figure 4.8: Another Example of The First Scenario.....	70
Figure 4.9: Simple Example About the First Scenario.....	71
Figure 4.10: Illustrate A General Example of Different Experiments.....	72
Figure 4.11: Second Scenario Setting.....	73
Figure 4.12: Illustrate The Changing in The Environment Complexity.....	76
Figure 4.13: Shows Another Example of The Second Scenario.....	77
Figure 4.14: Shows Example of Increasing the Complexity Of The Environment.....	78
Figure 4.15: Illustrates The General Path of The Second Scenario.....	79
Figure 4.16: The Proposed Experiment Works First Target Setting.....	80
Figure 4.17: New Targets Have Been Assigned.....	81
Figure 4.18: The Final Destination Have Been Reached By The Mobile Robot After Pick and Delivery Operation.....	84

Figure 4.19: Shows The Start State, Goal, And the Obstacle Position.....	86
Figure 4.20: Shows The Environment with The Moving Obstacle.....	87
Figure 4.21: Changing In the Environment Example.....	87
Figure 4.22: Overall System Experiment.....	88
Figure 5.1: Shows The Success Rate of Our Modified Algorithm Especially When Avoiding Obstacles In Path Trajectory.....	90
Figure 5.2: Shows The Success Rate of Avoiding Obstacles In Path Trajectory. These Results Show Before and After The Modifications And Improvements That Were Made.....	91
Figure 5.3: Shows The Number Average of Adding the Modification to The System.....	92
Figure 5.4: Modified CBS Algorithm Overview.....	95

ABBREVIATIONS

PL	:	Path planning
MAPF	:	Multi Agent Path Finding
MR	:	Mobile Robot
RRT	:	Rapidly Random Tree
A*	:	A Star
VD	:	Voronoi Diagram
GA	:	Genetic Algorithm
CPU	:	Central process unit
AI	:	Artificial Intelligent
CT	:	Conflict Tree
ANN	:	Artificial Neural Network
RNN	:	Recurrent Neural Network
CBS	:	Conflict Best Search
FL	:	Fuzzy Logic
UAV	:	Unmanned Aerial Vehicle
AMR	:	Autonomous Mobile Robot
AGV	:	Autonomous Guided Vehicle
DJ	:	Dijkstra's Algorithm
PRM	:	Probabilistic Roadmap Method

BIDRRT	:	Bi-Directional Rapidly Random Tree
PSO	:	Practical Swarm Optimization
PSBi-RRT	:	Practical Swarm Bi-Directional Rapidly Random Tree
EP	:	Extended Path
QL	:	Q-Learning
QT	:	Q- Table
BS	:	B-Spline
ML	:	Machine Learning
VRP	:	Vehicle Routing Problem
MAPD	:	Multi-Agent Pickup and Delivery
MARL	:	Multi-Agent Reinforcement Learning
MTSP	:	Multi-Depot Traveling Salesman Problem
MAPP	:	Multi Agent Path Planning
SIPP	:	Safe Interval Path Planning
DWA	:	Dynamic Window Approach
BFS	:	Best First Search
PCM	:	Pulse Code Modulation
SAC	:	Soft Actor-Critic
ACO	:	Ant Colony Optimization

LIST OF SYMBOLS

$[\]$	:	Integer Value
$[..]$	:	Array
$[...]^T$	:	Transpose Of Array
$ \cdot $	:	Absolute
$\in$	:	Belong To
Δ	:	Delta
$\wedge$	:	Value To Be Calculated
$\cong$	:	Almost Equals
$\forall$	:	All Elements
θ	:	Angle Of Direction
$\mathcal{N}$	:	Normal Distribution
$\mathcal{N}'$	:	Gaussian Membership Function
σ	:	Standard Deviation
μ	:	Mean Of Distribution
β	:	Band Width
c	:	Constant
dB	:	Decibel
ei	:	Ith Error

1. INTRODUCTION

1.1 NAVIGATION SYSTEM

Navigation systems are among those indispensable technologies that allow the precise determination of position and enable safe and efficient movement from one location to another. In recent times, modern applications have found wide usage in automotive, maritime, aerospace, and personal devices, impressing a marked impact on how people and organizations strategize and execute a journey [1]. A navigation system is supposed to be designed in a way that allows users to have the essential information about their location, destination, and the best available routes by taking the traffic flow, terrain factors, and various natural and artificial obstacles into consideration. Navigation in an outdoor modern setting is implemented based on GPS technology, supported by IMU and sensors for high accuracy and reliability [2]. With GPS, satellite signal-based, the possibility of global coverage with highly accurate location tracking is enabled, while IMUs add to this with speed and orientation. Besides, many navigation systems include mapping software that enables updating in real-time routes based on traffic or closed roads [3]. Navigation systems have experienced many changes, from the basic compass and map to the sophisticated applications that are user-friendly and residing on smartphones and in-vehicle installations [4]. These changes span from enhancing personal travel experiences to a change in how businesses operate vis-à-vis logistics and fleet management by route optimization for fuel efficiency and delivery times. Navigation technology also plays an important role in the autonomous operation of systems, such as drones and self-driving cars, since exact positioning and route optimization are of paramount importance with regard to safety and efficiency [5]. Navigation systems play a vital role in industries for positioning and route planning and have been upgrading themselves consistently on aspects of usability, efficiency, and safety concerning transport and logistics [6].

Navigation systems are those systems that enable users to find their way in various environments, using advanced technologies to improve pathfinding and situational awareness [7]. These systems utilize several components and elements, including sensors, processing units, and user interfaces, in bringing forth effective navigation solutions. Navigation can be defined as the system and integrated structure that allows an agent to

move from point A to point B by marrying localization, mapping, and path planning into a cohesive unit [8]. It enables various technologies like GPS, IMUs, and SLAM algorithms to establish the exact position of an agent onto a map or coordinate system when GPS signals are poor, or in an interior or urban canyon setup where GPS is essentially nil. High-precision localization usually adopts probabilistic methods such as Kalman and particle filters, constantly revising estimates of location by processing sensor input from LIDAR, radar, and cameras [9]. It is a methodology representing the environment in a fine-grained geographic manner, including semantic information such as roads, junctions, traffic lights, and pedestrian areas. SLAM may update the maps in dynamic conditions, [10] updates can be performed in real-time. The objective of the path planning is to find the optimal route using A*, D*-lite, and RRT algorithms; all these path-sampling methods using heuristic scores by distance, energy consumption, and estimated trip time [11]. These algorithms make real-time modifications and use sensor data in order to avoid unforeseen impediments in the form of people and other cars. Advanced systems use deep learning and reinforcement learning to increase the system's capacity for classification and interpretation of sensor data, moving object behavioural predictions, and policy updates based on prior experiences [12]. Further, combining localization, mapping, and path planning allows the navigation system to make such advanced adaptive decisions in dynamic real-world scenarios with a very high degree of autonomy, safety, and operational efficiency.

Systems such as those developed by Eder and Leonardi generate a Bounding Volume Hierarchies (BVH) while processing environmental geometry and returning optimized navigation paths [13]. User input should play the most critical role; hence, the system focuses the analysis process and data generated on user acts and current navigation state [14]. Lee's system uses radar and image processing to generate real-time dynamic maps of surroundings, identifying obstacles and accessible areas [15]. Meanwhile, Li and his colleagues propose the method using wireless signals for indicating the locations of mobile devices to assist route planning through electronic maps [16]. Su and co-authors describe a method using wheel speed sensors to calculate the position of a vehicle on a map with an eye toward more accurate navigation [17]. While these systems are huge strides in the field of navigation, challenges remain as far as making the systems become even more robust and as nearly accurate as possible in conditions when the signal availability is poor or when it is set in challenging terrains. The navigation system uses ambient volumetric parameters and

environmental analyses to create highly accurate paths, making it unique in enhancing the efficiency of navigation through different environments, including fixed and moving ones [18]. A navigation system includes an input unit for operations, a temporary storage unit, and a control unit processing navigation task using map data and application-related information.

1.2 MOBILE ROBOT PATH PLANNING

Robots basically play a very effective and important role in daily life in various fields, including industrial production, military operations, and relief in several cases resulting from disasters, as well as in places that are difficult to reach or that pose a danger to human life [19]. Among major aspects, robotics, and artificial intelligence include path planning. The essence of the process lies in the act of deciding on a feasible trajectory taken by an agent—such as a robot or an autonomous vehicle from a start point to a target destination by going around in the environment. This ranges from mobile robotics, autonomous vehicles, and aerial drones to even virtual character navigation in gaming and simulation environments [20]. In the main, the purpose of path planning is to determine an obstacle-free and optimal route that minimizes travel time, energy consumption, or other resource expenditures as shown in figure below.

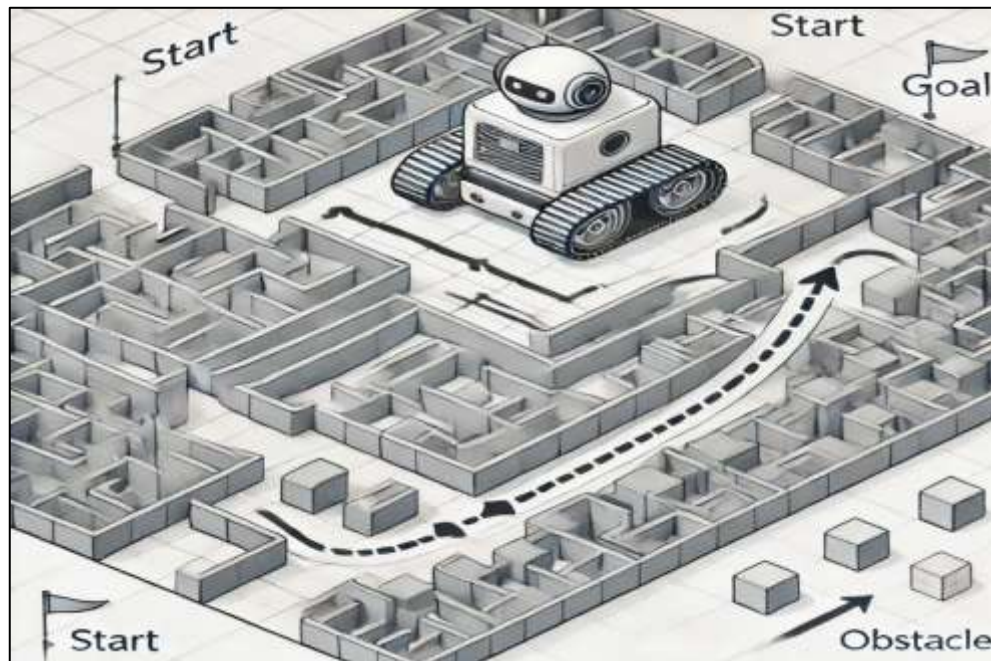


Figure 1.1: Path Planning Overview Example.

Depending on the environment's nature and application requirements, the approaches to path planning can be divided into a few concepts [21]. First, in the case of static environments, where the position of obstacles does not change, algorithms may make use of detailed maps and models for the computation of an efficient path [22]. On the other hand, dynamic environments-where obstacles may be moving or change over time-demand far more adaptive strategies fully capable of reacting to real-time conditions [23]. One of the most famous algorithms in path planning is A*, also called the A-star algorithm, which combines the merits of Dijkstra's algorithm with heuristic search [24]. A* finds the shortest path much more effectively by taking into account the cost of the path from the start node to the target with an estimated cost to reach the goal-known as a heuristic [25]. Other important algorithms include Rapidly exploring Random Trees, which work well in high-dimensional spaces, and the D* algorithm, developed for dynamic environments where paths may need to be updated continuously [26]. Adding to classical algorithms, some of the modern developments in path planning also involve the use of artificial intelligence and machine learning. For example, reinforcement learning can be put to work in order to enhance path planning through active learning of agents by gaining experience and thus adapting for navigation. This is hopeful in complex environments where traditional algorithms may struggle [27].

Challenges in path planning often involve how to get through such complex terrains, handle uncertainties in sensor data, and manage constraints such as vehicle dynamics or safety requirements [28]. In general, path planning algorithms should take into consideration issues like obstacle avoidance, smoothness of the path, and computational efficiency [29]. The integration of sensor technologies like LiDAR and computer vision has also further improved the capabilities of autonomous systems to perceive their environment, thus enabling more precise and responsive path planning [30]. This result is magnified when these technologies are fused with sophisticated algorithms that enable real-time navigation and obstacle avoidance, further greatly enhancing the effectiveness of path planning in practical applications [31]. In short, path planning has been one of the very significant research areas along with applications in the sector of robotics and AI [32]. This is also basically because capability enablement is substantially required for independent navigation as mentioned in [33]. With the progress of technology, development of more efficient, adaptive, and intelligent path planning algorithms shall continue to shape the future with autonomous

systems being driven through innovation in fields like transportation, logistics, and entertainment.

In addition to navigation, path planning has a great impact on efficiency and safety regarding the operations of autonomous systems [34]. A path-planning algorithm might be effective based on environmental complications, agent types, and mission goals. For example, in applications such as warehouse automation, robots have to operate in dynamic environments with moving obstacles, such as other robots or human workers, so that an algorithm would need to quickly adapt to changes [35]. In the case of flying independently, aerial drones, for instance, make altitude adjustments and follow airspace regulations in addition to path planning as mentioned in [36]. Another important field of research has to do with multi-agent path planning, where the problem to be solved is related to how to move several agents to avoid conflicts and optimize the general efficiency of the path. Some developed techniques involve potential fields-where virtual forces guide the agents around obstacles towards goals-or centralized and decentralized algorithms that manage agent-agent interactions [37]. As smart cities and connected environments emerge, path planning is increasingly set up in integration with traffic management systems that have real-time data feeds for better route optimization that reduces congestion. This holistic approach provides enhanced performance not only for individual agents but also in terms of system efficiency and safety in shared spaces.

In this regard, integration of path planning with decision-making frameworks and predictive analytics, as research in the field progresses, may hold the potential to revolutionize how autonomous agents operate in the real world and complex environments [38]. The process of planning paths is one of the basic pillars without which a robot cannot reach its goal accurately and in a short time. In order for robots to be able to advance and reach their goals, they must avoid several factors present in the environment, and acquiring this skill makes them distinctive and important [39]. Path planning algorithms for mobile robots in different environments play a fundamental role in avoiding obstacles present in the environment, whether known or unknown [40]. One of the main goals of path design is to create a path free of all collisions from the robot's starting point until it reaches its goal smoothly [41].

Path planning for robots refers to the possibility of safe access from the starting point to the destination. Path planning can be classified into global and local awareness [42]. This

research is focused on the path planning challenges of AMRs, considering their kinematic constraints, as often performance and responsiveness conflict. The contribution of the proposed work is to solve this problem in dynamic environments using a multi-agent policy learning approach. In this respect, it will integrate a path planning framework with a centered learning mechanism combined with decentralized execution that will meet both the performance and responsiveness criteria [43]. Global planning for robots is when the environment is known and provided with information about it in advance, whether the locations of targets or the locations of obstacles [44]. Compared to the local reality, the robot is supposed to pass through environments that may be unknown or unknown targets or obstacles. Therefore, most algorithms related to path planning fail to reach or collide with obstacles because they do not contain information that enables them to reach their targets or avoid obstacles [45]. Path planning is divided into two main categories, which include: global path planning and local path planning; each serving different purposes during navigation. Global path planning aims to find a complete trajectory from a start to a goal by making use of extensive prior knowledge of the environment, usually represented in a static map [46]. It finds an optimal path from the given criteria, whether it is distance or cost, using algorithms such as A* or Dijkstra's [47]. Local path planning, however, involves real-time navigation and the ability of an agent to change its course concerning immediate sensory input and dynamic conditions [48]. This planning is essential in effectively conducting motion planning in moving obstacle environments with powerful reactive strategies, such as those based on potential fields and DWA. In this context, global and local path planning together provide a strong basis for autonomous systems to efficiently and safely work their way through both static and dynamic settings as mentioned in [49]. In light of the developments witnessed by path planning, it has become necessary to pay attention to path planning in unknown environments to ensure smooth transition between obstacles until reaching the goal as shown in Figure 1.2.

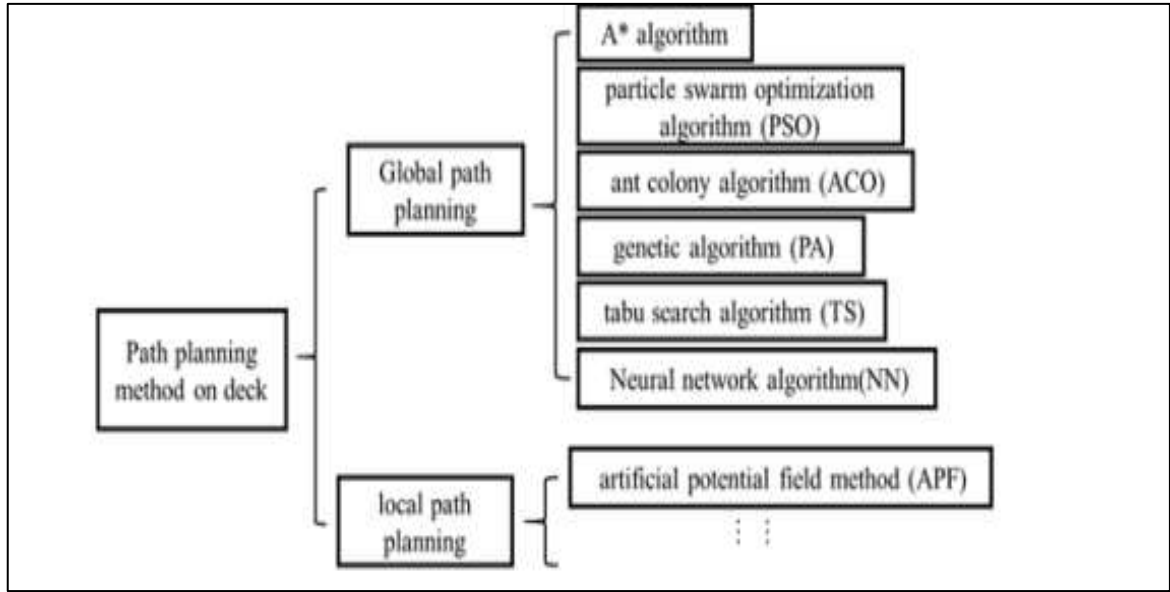


Figure 1.2: Global and Local Path Planning Classification [50].

1.3 MULTI-AGENT PATH PLANNING

Multi-Agent Pathfinding is a problem in which multiple autonomous agents need to find collision-free paths in some common environment that can be provided as a grid or graph [51]. Multi-Agent Pathfinding, or MAPF in short, from a robotics and artificial intelligence perspective, is an important area of research that faces and solves the problem of moving multiple autonomous agents through a shared environment collision-free. As the number of agents grows, computing near-optimal paths becomes exponentially harder; therefore, MAPF is a challenging combinatorial problem. The main objective of MAPF is to plan collision-free paths of all agents to get closer to destinations for minimizing the total cost involving travel time, energy consumption, or total distance [52]. MAPF finds essential applications across a wide spectrum, from robotic swarm motion and autonomous vehicles to warehouse automation and even gaming. Applications such as warehouse operations, where numerous robots will have to navigate through the environment for item carriage, require good path planning to maximize throughput while avoiding unnecessary delays. Similarly, in urban environments, autonomous vehicles will encounter congested roadways and must make their way safely and efficiently, with MAPF techniques being vital for traffic management and route optimization [53]. The two major approaches towards the MAPF problem are centralized and decentralized. Centralized algorithms compute paths for all agents simultaneously, leveraging global information for optimality and prevention of

conflict. Those often rely on search-based methods, graph-based approaches, or optimization algorithms. Decentralized algorithms enable independent agent decisions based on local information; these can scale to large systems but are usually achieved at the cost of optimality and efficiency. Other challenges in MAPF involve the handling of dynamic environments, wherein obstacles may suddenly appear, and optimizations for a wide range of objectives, from minimizing the total travel time to efficiency maximization or energy consumption reduction. Besides, real-time constraints require developing algorithms that can cope efficiently with changes in an environment so that agents will be able to navigate safely and efficiently.

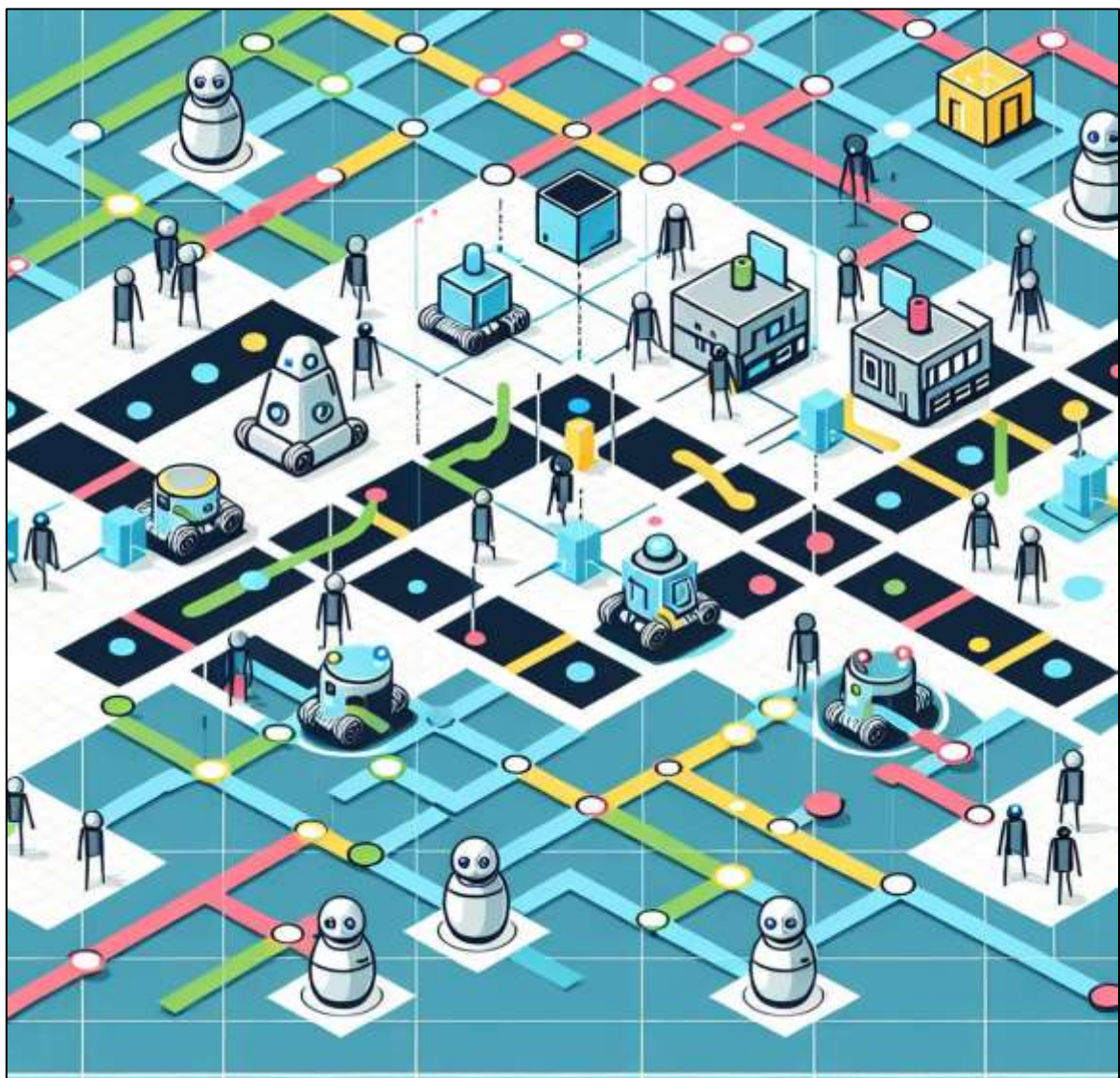


Figure 1.3: MAPF Overview Example.

Considering that MAPF is still an evolving area of research, recent works use most of the techniques developed within artificial intelligence, such as reinforcement learning and cooperative strategies. In particular, these developments will make the ways in which autonomous agents navigate shared spaces in smarter, more coordinated, and efficient ways possible for a wide range of applications. In other words, Multi-Agent Pathfinding is a really interesting area that encompasses the research contribution to finding an effective solution for autonomous agents to be able to move in shared environments. The developed algorithms and strategies should be such as to further optimize agent motion and enable safer and more effective operations in a wide range of practical applications, where each agent needs to find its way from some start position to some goal position. The challenge is not only in establishing safety through the prevention of collisions but also in the best pathfinding optimization according to some criteria that include minimizing travel time or path length [11]. MAPF finds many interesting applications, such as robotics, where this problem allows effective coordination of robots in some tasks like warehouse automation, traffic management, and gaming. As the number of agents increases, or the complexity of the environment, the problem of MAPF will become increasingly difficult to solve optimally and scalable using more sophisticated algorithms [12-17]. Multi-agent path planning or MAPF is a complex problem that involves coordinating several agents in moving from their initial locations to their targets while avoiding collisions [17-22]. It is an important problem with large practical interest due to its applications in domains like robotics and logistics. Here are some key approaches and observations about MAPF: One method employed is the Push and Swap Algorithm, which seeks to mitigate conflicts by deconstructing the issue into a series of individual route planning tasks executed sequentially. Nevertheless, it is not complete in every case, and there is a need to develop a more reliable solution called Push and Rotate, which assures completeness under particular conditions and runs in polynomial time [24]. Another evolution of MAPF is continuous time pathfinding, whereby an MAPF algorithm combines SIPP with CBS to work within a continuous time domain hence providing optimal solutions without the limitations due to discrete timesteps [25].

1.4 STATIC ENVIRONMENT OVERVIEW

Static environment will be able to define a situation where its defining variables do not change over time. The concept has wide applications in almost all disciplines, and it really

allows people to understand systems and their relationships better, void of clutter provided by the dynamics. In this case, the base elements stand still; hence, it would be highly predictable and model behaviors rather precisely [54]. One of the distinguishing features of the static environment is stability and predictability. This predictability offers a setting in which researchers or analysts can get predictable results for a specific set of rules or equations. For example, a controlled experiment where all the environmental variables-from temperature to pressure-remain constant will facilitate the scientists to study with absolute surety the effects of specific changes in one variable while the rest of the variables will not intervene [55]. Aside from that, static environments are predictable due to a lack of influence from the external environment. In business and economics for instance, a static environment is one where all market conditions do not change. Firms can then evaluate strategies based on particular costs and benefits as well as consumer behavior, all of which are invariant [56]. This simplification might come with the potential for less complex decision-making processes since the concerned parties do not need to worry about changing market conditions. In software engineering, of course, static environments are extremely useful in the design and testing phases. Applications designed for static scenarios often need only a few updates or changes, and the systems are strong enough to function well under the predefined circumstances. For instance, a database management system that is designed to operate under a fixed load of transactions can be tuned to work efficiently without too many complications caused by diverse user inputs or volumes of transactions [57]. Static environments are only part of this modeling-a population within a closed ecosystem through which no entry or exit of new species occurs, and the environmental conditions remain constant. With that, researchers can look into only the interactions between species and their habitat, without needing to bring in external pressures as factors that might alter the dynamics of the ecosystem. This kind of modeling helps one understand population stability, resource competition, and the effects of environmental changes in a controlled manner [58]. Application can also be found by static environments in mathematical modeling and simulation. Most of the optimization problems researchers usually model assume constant parameters in order to derive solutions that can then be applied under certain conditions. Such models enable a wide range of analyses, from financial forecasting to logistical planning, with negligible effects of variable change.

The static environment may have implications for policy formulation, say, in the management of the environment [59]. To this end, an analysis of fixed conditions would allow the policymaker to devise focused, strategic solutions that meet present needs without the complicating factor of considering how it would look in a probable future. This could be more informative while dealing with resource distribution, conservation, and regulation to safeguard environmental sustainability. Although the static environments indeed offer a good analytical framework, there are some shortcomings: most real systems often depend on so many dynamic factors, therefore displaying changes in magnitude that may affect the results [60]. But while the static models are clear and simple, it may not fully capture the complexities of ever-changing environments. As a result, it becomes of increasing importance for the researcher to consider how differently static assumptions correlate with real-world conditions at the formulation of strategies from such models. Static models use the frameworks necessary for an understanding of systems in numerous fields. The nature of stability, predictability, and lack of any external influence encourages accurate modeling, effective decision-making, and also strategic planning. Such static models allow the analyst and the policymakers to acquire key insights necessary for the guidance of actions and activities within the realms of the limitations set forth by the dynamic nature of the real-world scenario.

1.5 DYNAMIC ENVIRONMENTS OVERVIEW

Dynamic environments mean that the conditions might change over real, possibly unpredictable time, with moving obstacles, changing terrain, or other user demands. In general, the most important challenges to navigation and path planning involve dynamic environments, especially for independent systems like robots, drones, and self-driving cars [61]. While in static ones, the position of obstacles and other variables remains the same, in dynamic environments, it is necessary to make continuous changes at every instant in real time. One of the major problems affecting dynamic environments involves moving obstacles [62]. Pedestrians, cyclists, and other vehicles may enter the path of an autonomous vehicle in urban settings and thus require immediate adjustments from the planned trajectory to avoid collisions. In warehouse automation, mobile robots must move around human workers or other robots who have irreconcilable changes in their paths, hence complicating finding a path [63]. Advanced sensor data and real-time processing are generally called for to integrate

into most autonomous systems for operation in dynamic operating environments. LiDAR and radar basically provide an agent with the ability to perceive its surroundings, including changes detected by using computer vision, while algorithms for dynamic path planning update the optimum route of this agent based on the information obtained most recently [64]. The techniques to be used in such environments for ensuring safety and efficiency of operation include local path planning, which focuses on near-future navigation challenges, and techniques of multi-agent coordination as shown in Figure 1.4.



Figure 1.4: Dynamic Path Planning Example.

Dynamic environments can also exhibit various types of uncertainty. Sometimes, partial information about the environment, such as the existence or absence of some obstacles, or the actual conditions of the terrain, might be known. Because of this, dynamic scenarios often involve incorporating probabilistic methods into path-planning algorithms, which enable them to estimate the likelihood of certain events occurring and make appropriate decisions in the face of uncertainty. In general, dynamic environments pose a big challenge in autonomous navigation due to the nature of objects and movement of obstacles within the environment being usually unpredictable [65]. The ability to navigate successfully within such an environment will require not only highly advanced sensing technologies but also adaptive algorithms that update the system in real time and maintain the safety and efficiency for any autonomous system operating within that environment. As research in this arena continues to develop, more intelligent and robust navigation development will assist in deploying the agents autonomously in dynamic and unpredictable environments.

Dynamic environments introduce more complexity and challenges. For instance, while studies show that such an environment's complexity can be NP-hard, some structural constraints on specific instances can make the problem solvable in polynomial time for a fixed number of agents [66]. Graph-based approaches form the MAPF as a satisfiability problem on unit-distance collision-free graphs, hence underlining the applicability of graph theory in solving the problem of MAPF [67]. Despite progress, there are still challenges, especially in dynamic settings and achieving optimal solutions in various scenarios. Future research may concentrate on improving algorithm efficiency and adaptability to real-world applications. Within the framework of Multi-Agent Pathfinding (MAPF), a variety of mobile robotic entities are employed, each possessing distinct traits and utilization contexts [38]. Autonomous Guided Vehicles (AGVs) alongside Autonomous Mobile Robots (AMRs) are frequently deployed in industrial environments, including warehouses and manufacturing facilities, where they maneuver within communal areas to convey products while preventing collisions [39]. Conversely, drones, also known as Unmanned Aerial Vehicles (UAVs), function within three-dimensional airspace, necessitating meticulous coordination in MAPF to avert aerial collisions [40]. Swarm robots, consisting of many simple robots acting together, introduce the MAPF challenge that there will be very many in number, and all moving in a coordinated manner [41]. Humanoid robots are designed to walk like humans in spaces made for humans-for instance, factories or public areas-and therefore have to solve

the MAPF problem to avoid interference. Furthermore, there are AUVs operating in the aquatic environment, and MAPF is very important for generating collision-free paths in the exploration or monitoring missions [42]. Each type of mobile robot has different MAPF challenges depending on the type of environment and the type of mission to perform.

1.6 CHALLENGES AMONG PATH PLANNING

The basis of path planning in robotics, artificial intelligence, and autonomous systems is to find the most feasible route for agents from a starting point to a desired destination, taking into consideration that it should not interfere with any other agent or obstacle. Despite major developments within path-planning algorithms and techniques, several problems persist, which complicate the development and application of efficient pathfinding solutions [68]. Recognizing such challenges could very well improve existing algorithms by developing newer methodologies concerning specific applications and environments. Among the main difficulties of path planning is environmental complexity. Realistic environments are usually very complex in terms of the presence of different obstacles, varied terrain, and dynamic aspects. For instance, the aspect of landscape might change dramatically with moving obstacles like pedestrians or other vehicles and at times paths need to be updated in real time [69]. Environments can be very structured, starting from cluttered indoor environments to open outdoor areas, each having their unique strategies for planning. Similarly, motions by path planning algorithms should be flexible; that is, they should efficiently move in complicated, unstructured spaces and ensure safe motion. Another important challenge regards the computational efficiency of path planning algorithms. For instance, if either the number of agents increases or the detail level of the environment is higher, computation required to evaluate all possible paths may rise exponentially as mentioned in [70]. This is the "curse of dimensionality," where the size of the search space becomes so large that finding an optimal path in any reasonable amount of time becomes computationally infeasible. Hence, algorithms must balance optimality with efficiency and have led to exploration into the use of heuristics, approximations, and many optimization techniques to accelerate the process of pathfinding.

Dynamic environments introduce further challenges into the path planning process. Unlike static environments, where obstacles and conditions are fixed, dynamic environments pose a situation where agents need to constantly update their paths according to real-world

dynamics [71]. The need to respond to unforeseen events-such as newly appearing obstacles or other agents changing their trajectories-motivates the use of robust local planning strategies. Furthermore, uncertainty in the environment, whether due to incomplete information or sensor noise, can make it impossible to effectively decide on a path and may call for probabilistic methods of path planning. Multi-agent coordination adds another layer of difficulty to the problem of path planning: when several autonomous agents work in the same environment, ensuring conflict-free motion of the agents can be a fundamental requirement [72]. Agents need to coordinate with each other in such a case to avoid any collisions and further assure the optimized performance of all agents en masse. This can include developing algorithms which consider not only individual agent paths but also interactions between them and the global state of the environment, for example in applications like warehouse automation or traffic management. Finally, there is the challenge of real-time constraints in path planning. For many applications, especially for safety-critical applications such as autonomous vehicles and robotic surgery, decisions should be made with as little delay as possible, which can ensure the safety and efficiency of operations. In turn, it means that algorithms have to process large volumes of data and give a workable path in milliseconds; thus, they have to be highly optimized and responsive. Path planning challenges are multifaceted, touching on aspects of environmental complexity, computational efficiency, dynamic conditions, coordination of multi-agents, and real-time requirements. Any attempt toward addressing these challenges has to result from an integrated approach: algorithm advancements coupled with the advancement of sensor technologies and artificial intelligence. Further research in this area should make a way for developing sophisticated and adaptive solutions for path planning, which is so critical in the successful deployment of autonomous systems in various applications as shown in Figure 1.5.

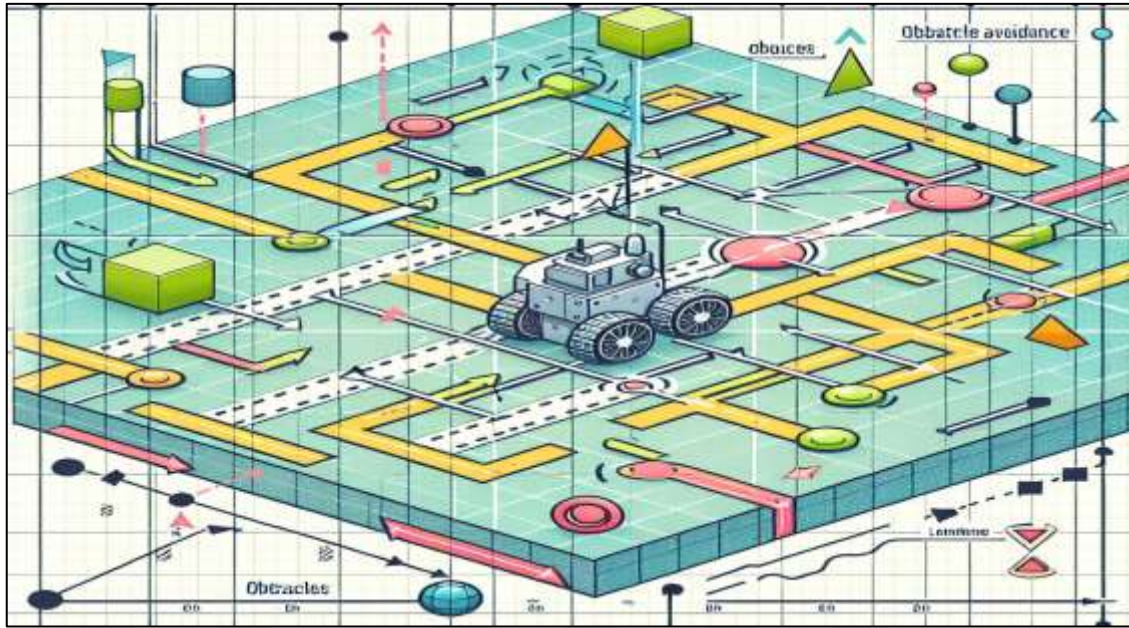


Figure 1.5: Obstacle Avoidance Challenges.

One of the challenges facing path planning for multirobot is how to find collision-free paths in an environment containing a set of fixed and moving obstacles. Optimizing travel time, path planning, and distance taken to reach are essential tasks facing robots. They must move from the starting point to reach the goal without collision, and here complexity arises, and the problem becomes greater due to the presence of several factors within the environment that can affect path planning. Ensuring safety and work efficiency requires relying on algorithms that have the ability to plan paths quickly and accurately and deal with changes that occur in the environment without any delay, especially in time and decision-making [73]. This requirement represents a real balanced and an effective challenge for this type of problem. Mobile robots are gaining recognition and being employed more in practical situations. While singular, autonomous robots have great merit, it can be very advantageous, in certain scenarios, to deploy several at a time. There is, however, one huge hindrance regarding this within the realm of robotics: how to efficiently coordinate many robots working amongst one another in fluid, uncontrolled conditions. One of the best ways of dealing with this issue is to apply the principles of Multi-Agent Path Finding.

The MAPF problem is a search for collision-free paths of multiple acting agents in the shared environment. This topic has been extensively studied, and many algorithms have been developed so far, addressing different real-life situations. Such systems will find wide

applications, especially in today's world with growing automation of tasks, including package delivery. A methodology has been established for managing multi-robot systems that resemble automobiles, where automated vehicles are a subject of active research and, in some cases, are implemented in practical environments. Also included are autonomous taxis [74]. In addition, these technologies have their applications in office robotics, unmanned surface vessels, and video game AI. In games, a single player often needs to manage large numbers of heterogeneous agents in dynamic or crowded environments, which is commonly required in genres like turn-based and real-time strategy. Also, in applications of automated warehousing, which generally consist of integrating a fleet of robots for automating tasks related to product storage and sorting. MAPF has achieved much importance, in these domains the first challenge comes from highly compact settings, which require operating near-optimally with a large fleet of robots. Algorithms designed for these settings must manage the continuous assignment of new tasks to robots following the completion of their previous assignments. This therefore calls for the need to develop mechanisms that would be able to manage large fleets with high levels of productivity efficiently [74-79].

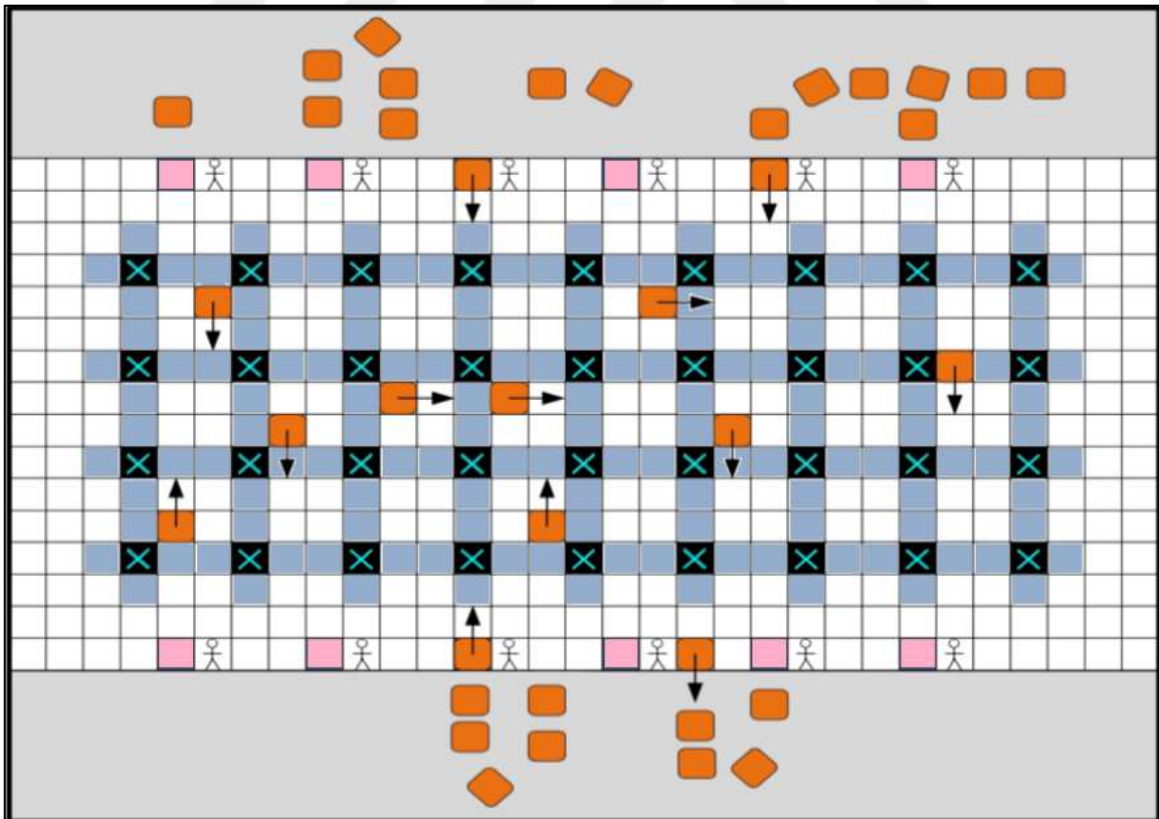


Figure 1.6: Warehouse Path Planning Example of Pickup and Delivery Operation [75].

Figure 1.6 shows an example of multi-agent inside warehouse environment. Despite the diversity of scenarios under which the MAPF problem can be applied, existing techniques almost always sacrifice certain aspects for realism or computational efficiency. To make the problem more tractable, considerations such as kinematics, failures, and uncertainty are usually ignored when executed in real time [76]. Most algorithms obtain plans that later have to be modified when running on real robots. The more abstract the plan is, therefore, the more changes it will require to work correctly. In this respect, formulating strategies which require little modification but are still effective might be one resourceful way.

1.7 MOTIVATION

Implementing MAPF algorithms in real-world settings can be more challenging than solving the problem in a controlled, theoretical environment. Several factors are often overlooked in the pursuit of practicality when designing robot paths. For instance, imagine two robots moving from their starting points to their intended destinations in the real world. In this, the paths are intentionally drawn as curved, representing real-life conditions more accurately, while color-coding represents each robot. A MAPF was used to solve this problem and returned that a straight line would, in fact, bring the robots closer together with no collisions.

1.8 OBJECTIVES

The main goal is to find a set of paths and plans that can be realized safely with no a priori unknown conditions while keeping the work efficiency. In this thesis, we introduce a new generation algorithm, multiple goals, time division, and a priority-based approach. The generation mechanism will be capable of running with a high success rate in achieving objectives under most unpredictable environments, the summarization of the objectives point will be as:

- a. The work structure is summarized in planning within environments that are divided into two types: simple in structure and complex in structure, according to the type of environment and the types of obstacles within this environment. It is divided into known and unknown environments.

- b. When a change occurs in the work environment, the change in planning must be smooth without affecting the success rate of planning. Here comes the power of the proposed method.
- c. Achieve high success rates while avoiding obstacles, achieving high success rates depends on the percentage of overcoming obstacles within the environment until reaching the goal.
- d. Solve the shortest path problem in an efficient manner. The process of choosing the shortest path plays a fundamental role in planning routes as it has a significant impact on the rates of customers reaching their goals. Also, successful planning within moving environments sometimes makes the matter difficult, but with the presence of real impact factors, planning becomes successful and achieves successful arrival by the shortest path.
- e. Multi-agent systems operate to handle multiple goals. Planning routes for multiple targets plays a fundamental role in important locations such as factories or warehouses, so proper planning leads to accurate results. Planning for multiple targets makes the work complicated, but after adding several criteria, this issue was achieved smoothly.

1.9 CONTRIBUTION

A conflict-based search (CBS) algorithm has been developed to facilitate the planning of robot trajectories in a range of environments. The method presented shows significant efficiency in various situations, encompassing dynamic maps, obstacles, and additional environmental influences. It provides comprehensive flexibility in improving goal search results and adjusting to changes in the environment without necessitating any prior knowledge of the surrounding conditions. While many dynamic pathfinding methodologies are performing well in the case of small changes on the map, regular changes lead to long time for adjusting paths and even more possibility of a pathfinding failure. Sudden change of a map may affect efficiency adversely, especially in large-sized environments with lots of mobile obstacles. This work is devoted to the development of such a path-planning approach that will give agents the ability to perform movement with the avoidance of obstacles of different shapes, such as concave, rectangular, and circular forms. Present a modified method for CBS in robot path planning within mixed environments. This independently executes a local route planning by selecting the best route path through a prejudging process to reassess the neighbouring areas in order to avoid obstructions. The approach embeds robot

waiting and circuitous routing. Comparisons with relevant prior work were made to validate the proposed method. This study comprehensively defined the critical elements that are involved in the creation and fine-tuning of multi-robot pathways. In the experimental results, it is shown that the approach proposed in this work has achieved a high success rate in avoiding obstacles and arriving at goals in record time.

1.10 ORGANIZATION

Chapter 1: This chapter presents the main introduction to our thesis. It contains also the work remarks as motivation, objectives, and contribution.

Chapter 2: This thesis takes a step back and surveys the landscape, dividing into categories defined by common characteristics and approaches. The discussion emphasizes how the representations of the environment differ, but also that a graph representation in the abstract enables reasoning about optimality and completeness. Non-optimal algorithms are also compared, including rule-based solvers. Then comes the chapter about optimal solvers, we describe how from the most general methods of MAPF solving one ends up with the CBS framework. We analyse our main competitor: the Modified Conflict-Based Search (CBS) approach.

Chapter 3: This chapter deals with the basic concepts needed to understand the CBS algorithm. We study the structure of the planar network and its simplest paths, called Non-Complex paths, by making use of pathfinding algorithms. Utilizing these latter, we define a theoretical framework which will represent the basis of the path-planning algorithm. Finally, we build up the path-planning algorithm using those mathematical premises, proving optimality and completeness.

Chapter 4: After describing the mathematical background of the algorithm, we will follow up with its implementation aspects by providing flowcharts for the major components. We will also describe the structures forming building blocks of the algorithm, which are useful later while deriving its complexity.

Chapter 5: It presents the results obtained from simulations, along with an analysis of the work scenarios and the corresponding outcomes.

Chapter 6: It presents the conclusions and future work.

2. RELATED WORKS

2.1 SEARCH STRUCTURE

The search structure is an essential data structure in computer science and AI that helps in arranging information efficiently to access the data from a collection. They are frameworks with a definite procedure on how to explore data in order for algorithms to accomplish tasks of locating desired elements or solve a complex problem in efficiency. Performing search structures can be done with arrays and linked lists, which are in linear structure, and can also be in a tree or graph, which is more of a hierarchical structure. The essence of any search structure would be to search in an optimal manner, that is, minimize all kinds of time complexities to retrieve data. Using a linear approach, one would have to check every other element, while with an advanced data structure such as a binary search tree, it could achieve logarithmic time complexity by maintaining a sorted order of elements. Graph-based models make it very much convenient to research relationships and pathways between nodes, due to which they find their prominent implementation in networking, pathfinding, and social networks. Different search structures are suited for various types of data and problem domains. Hash tables, for instance, provide particularly good facilities to store key-value pairs and then look up those values given a certain key. Thus, they are appropriate for applications needing to look up quickly stored elements. Priority queues, on the other hand, are relevant to work-scheduling or resource-allocation problems that need to process elements in priority order. Search structures, in particular, are important for both data organization and retrieval. The efficacy of algorithms, to say the least, does depend upon the underlying structure. Because design and implementation affect application performance, choosing the right structure is important.

In this work, we first begin by highlighting some of the elemental problems related to pathfinding and critically analysing the literature approaches to handle these issues. Then focusing on a grid-based pathfinding problem through a systematic process of finding out, elaboration, appraisals on the nature of inherent problems, and finding the potential solutions for the same. Setting of Multi-Agent Path Finding, where those real-world complexities make these issues much worse. These are complexities that comprise the main and most

challenging parts of our work, in that sense emphasizing their key role in moving the field forward as shown in Figure 2.1.

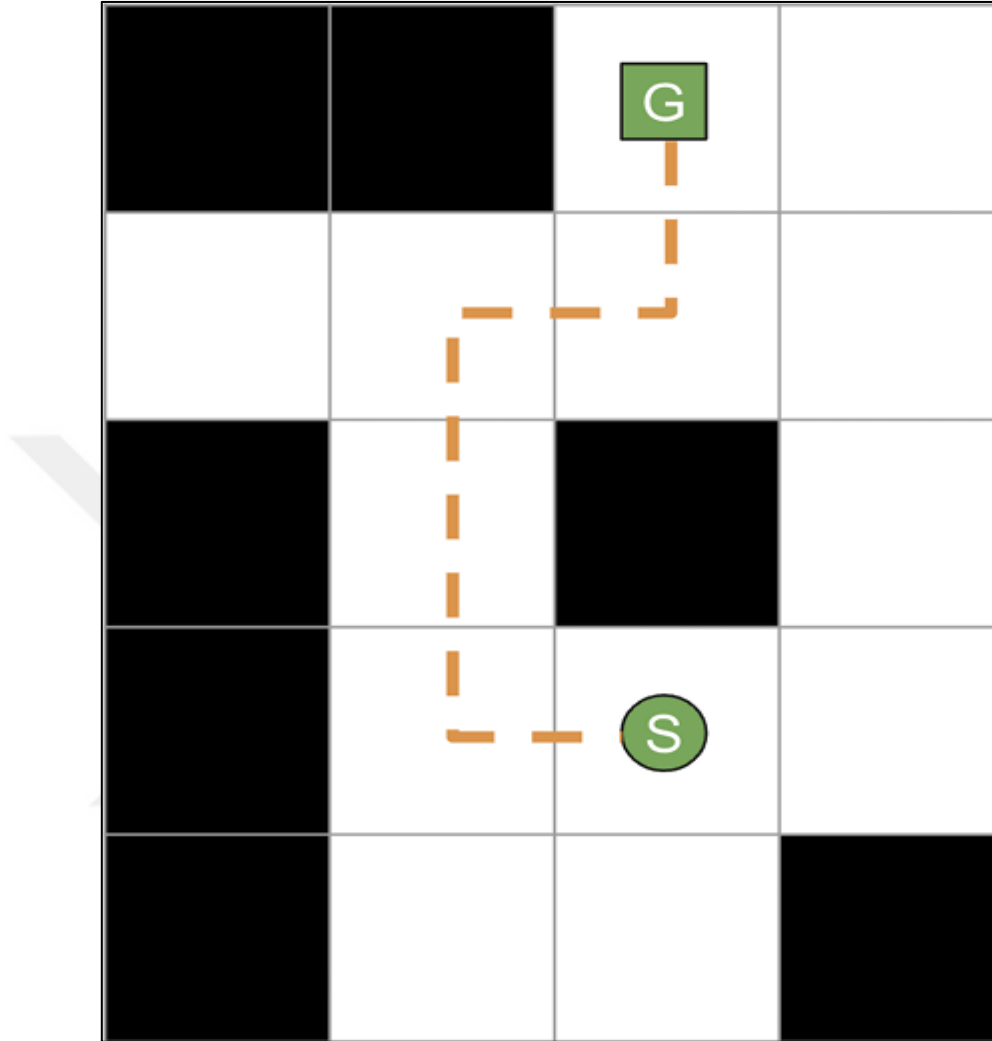


Figure 2.1: Shows An Example of The Start State and Goal State Inside the Environments [76].

Excessive turning points, sharp turning angles, and turning points too close to obstacles are common in classic A* algorithms. All these factors have higher chances of increasing energy consumption, complicating the motion control for Automated Guided Vehicles, and raising collision risks, especially when turning points become crowded around an obstacle. In this respect, an improved A*-based path planning algorithm is proposed as in [77]. AGV intelligent parking provides an effective way to solve the problems caused by urban parking. In an AGV parking lot, the rationality of the running path of the AGV determines the overall transportation efficiency. In the path planning of AGVs, the RRT algorithm and ACO algorithm have been proved to be effective. However, dependence on uniformly distributed

initial pheromones may result in blind searching and local optimization during the early stage of the algorithm. Finally, it reduces the efficiency of AGV [78].

2.2 GRAPH SEARCH STRUCTURE

Graph search structures are fundamental building blocks for Multi-Agent Pathfinding, which aims at finding optimal paths efficiently for several agents navigating through a shared environment. MAPF essentially means coordinating the movement of multiple agents from independent starting positions toward specified goal positions without causing collisions either among agents or with obstacles. Environments are typically modelled as graphs, where vertices represent valid positions and edges represent possible movements between those positions. The challenge of MAPF lies in the complexity due to the use of several agents operating in the same area. The path of every agent might interfere with others, creating some conflicts that need to be resolved so that all agents reach their goals safely and efficiently [79]. This could be done, for instance, by implementing such strategies that are able to foresee and avoid conflicts. Examples include priority-based methods or the use of flow-based techniques. Similarly, in MAPF, search structures are also used to model the state of the environment. It includes not only the position of agents but also the configuration of obstacles and previously traversed paths. Some of the techniques used, depending on the specificities of an application, are centralized search - an algorithm computes the paths of all agents at once - and decentralized search - agents operate independently but communicate their plans. Advanced techniques rely on graph search structures, like Cooperative Pathfinding, in order to achieve better efficiency of agent movement. Most of these methods apply heuristic optimizations to keep the computational load low while ensuring that the solution is collision-free. With graph search algorithms and using adequate search structures, MAPF can handle multi-agent coordination efficiently; hence, it is an important area of research in robotics, game development, and logistics.

Graph search structures generally form the backbone of Multi-Agent Pathfinding to enable the efficient computation of multiple agent paths in a shared environment. Several search algorithms combined with different strategies form the backbone of MAPF to address the challenges posed by agent-agent interactions and collision avoidance. By doing so, it allows all agents to navigate their environments effectively while minimizing potential conflicts [80]. It is expected that as the field evolves, further work in terms of implementation of

sophisticated search structures and advanced algorithms will continue to be done in enhancing the capability of multi-agent systems. A multi-agent pathfinding (MAPF) problem can be defined using a graph $G = (V, E)$ and a set of agents $\{a_1, \dots, a_k\}$. Time is discretized into steps, and each agent can either wait at its current vertex or move to a neighbouring vertex in subsequent time steps. Waiting incurs a unit cost unless an agent is at the final goal. The AI must decide between moving or waiting based on the action that will be taken from the start position. Node conflicts arise when agents a_i and a_j occupy the same vertex (u, v) at time step t , potentially increasing the total cost of the paths[49-59]. In a dynamic warehouse challenge involving multiple robots, there is a predefined configuration and specific target positions. The problem can be viewed as an optimization task for robots to find routes that avoid collisions and allow each robot to reach its destination efficiently. Path planning in a multi-robot system requires the creation of collision-free paths with minimal robotic movement. The environment includes static obstacles that robots cannot bypass. The target mission is crucial, as the objective area provides essential information for direction planning. Generally, a multi-robot trajectory planning system involves mapping out goals for each robot to reach. The destination itself becomes an additional obstacle that impacts the final path calculation. Effective goal assignment aims to minimize the length of future paths, and the computation time required. Randomly assigning targets can significantly increase both computation time and path length. Therefore, it is preferable to employ a strategy that considers path length in advance. A graph with states from A to G, their weights, and directions. The red arrows in the figure represent the shortest path from start state A to goal state G. Here, the shortest path in the graph search problem is considered as one which develops the minimum cost-to-come, that is represented as $g(G)$. The cost-to-come can be defined as the summation of costs of the path from the starting state to any intermediate state. Through the plan, we can also understand the mechanism for choosing the shortest path through which we can reach the goal with ease, consider the example shown in Figure 2.2.

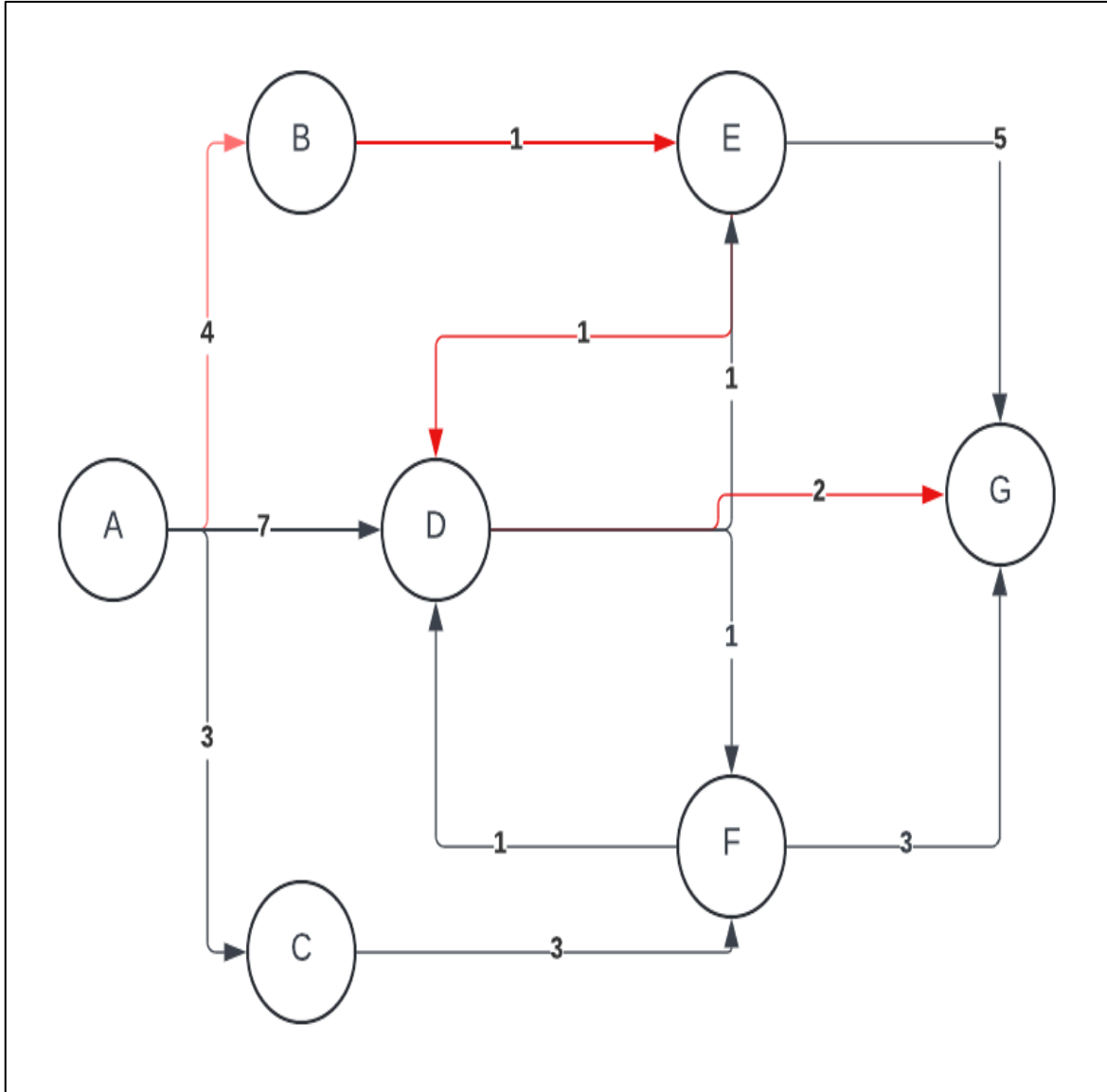


Figure 2.2: Shows An Example of Shortest Path Length.

2.2.1 Path Planning Methods Within Graph Search

Dijkstra's Algorithm [60-66] is one of the pioneering techniques in the area of shortest path computation. This technique maintains three kinds of nodes, namely, nodes to be processed, nodes in consideration, and nodes whose minimum distance has already been computed. Each node in this algorithm is marked with the minimum distance from the source node. Hence, it can then get the shortest path easily. A hierarchical Q-learning path planning architecture was proposed by [27] were provides a solution to the cooperative tracking control problem of MASs subjected to lumped uncertainties in an unknown environment. The hierarchy in the architecture of the proposed HQPP is divided into three layers: the

decision layer, the distributed estimation layer, and the local control layer. In this regard, a dynamic parameter and trajectory fitting Q-learning algorithm is proposed within the decision layer, allowing it to identify a viable continuous trajectory towards the target within an unknown environment. One of the key benefits with Dijkstra's Algorithm is that it is simple; it only requires the knowledge of the distances between nodes in order to be effective. A* [67] is one of the most used pathfinding algorithms, it was explored by [81] according to Dijkstra's algorithm, it incorporates a heuristic function, denoted by $h(x)$, which is about the estimation distance toward the goal state. For this heuristic function, there is an important condition. In no case, can it overestimate the distance to the goal. When this condition is satisfied, the heuristic function will guide the algorithm in selecting which nodes to expand and further explore, preserving optimality of the resulting path.

RRT algorithm, proposed in 1998 [10, 70], starts with a root node; it makes use of a random distribution function that samples child nodes within a previously defined map. Collision detection adds nodes that correspond to certain criteria in the random tree and expands it iteratively until it reaches the goal. Once the tree is fully created, the algorithm finds a collision-free path from start to goal within this generated tree. In [82], to solve the RRT problems of long path planning time and existence of a great number of redundant points, a new algorithm is proposed which introduces a parent point priority decision strategy and a real-time optimization strategy. First, to save time in path planning, the generation of a new point considers its parent beforehand in order to avoid traveling along the random tree in search of a parent point. Second, there is real-time optimization where distances of a newly generated point, its parent point, and two ancestor points from the target point are compared. This comparison will enable us to select a new point that elongates the random tree in such a way that only a minimal number of useless points result. Due to simplicity, flexibility, and efficiency for real-time path planning, the RRT algorithm enjoyed considerable attention. However, its reliance on random spatial growth often causes the algorithm to suffer from its reduced directionality, which can hence lower both efficiency and quality of the paths [83]. A number of enhancement strategies were proposed to overcome various limitations of the basic RRT algorithm. These include guiding the growth of the random tree with the use of obstacles and target points information, techniques for bi-directional expansion, such as Bi-RRT, and the performance of some optimization on the generated tree, using approaches such as re-selection of wiring nodes' parents, as in RRT*.

The blind search mechanism of the traditional RRT algorithm causes a lack of efficiency and computation intensity. Researchers have proposed such ways to incorporate guidance information like APFs [72-85], obstacles, and target locations [86] into the search space. This incorporation of APFs enables the algorithm to make its way to the target efficiently while avoiding obstacles, thus greatly improving the efficiency of path planning and enhancing the convergence speed of the algorithm. In this respect, the evolution of the RRT algorithm reflects how specialized optimization techniques were applied to specific challenges associated with pathfinding. The Bi-RRT method as in [87] makes the search for a path more effective and reduces the number of iterations to find a solution. Nevertheless, it has some limitation regarding the quality of the solution at the initial point and with respect to the convergence time toward the optimality. Such limitations have led the researchers to develop the PSBi-RRT algorithm [88]. This advanced technique incorporates kinematic optimization, postural evaluation, and goal-oriented sampling to reduce possible collisions in complex spaces. It further integrates node rectification methods, θ -cut, and node connections using triangular inequality in order to make the planned paths more accurate and optimal.

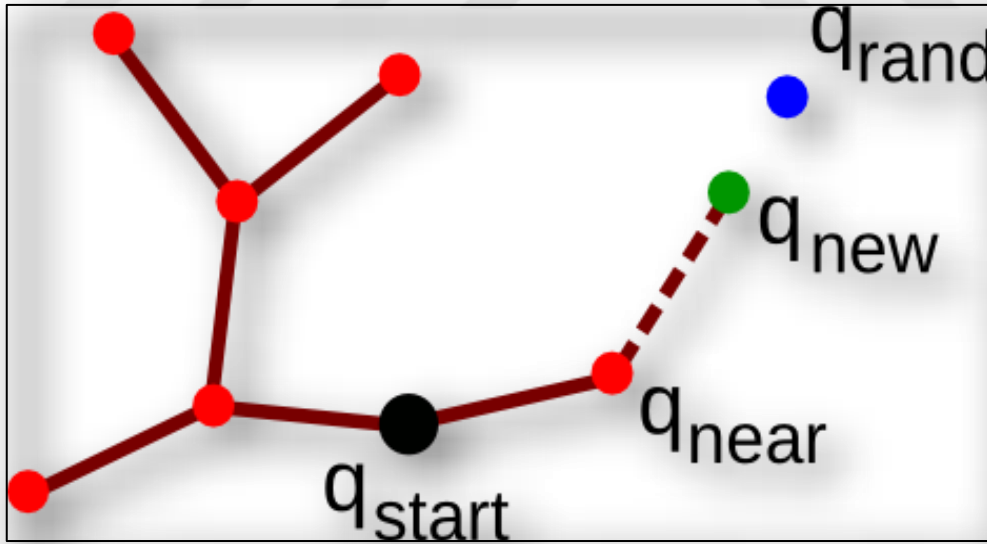


Figure 2.3: RRT Algorithm Structure.

RRT* as in [85-89] is an enhanced form of the RRT algorithm that, by applying techniques such as the re-selection of parent nodes and rewiring, reduces node cost, hence making it

more efficient in pathfinding. This further helps to make path searches more predictable. Combination greedy heuristic sampling with RRT-Connect does so much in the efficiency of sampling that there is minimal repetition of unnecessary samples [90]. A critical heuristic sampling involves determining the area of expansion. Ding et al. [91-93] thus proposed an extension to this approach, called Extended Path RRT*, or EP-RRT*. In the method, emphasis has been placed on extending the area of the path itself for rapid exploration of the surroundings to locate a valid path and thereby achieve an efficient heuristic sampling. In this respect, this algorithm has been further optimized by several authors: using an elliptical range for the sampling points [94], using curves instead of straight paths to have better obstacle avoidance capability [95-105], making use of environmental feedback for correcting deviations [106], and real-time updating of the starting point in the update cycle [102]. These are some improvements that allow faster convergence of this algorithm. The RRT algorithm has evolved by the inclusion of different algorithms and dynamic step techniques, which have enhanced the path planning performance in complicated environments key findings: 26. In the introduction of potential field functions into RRT, the P-RRT algorithm came up, which managed to reduce the number of iterations and faster convergence speed.

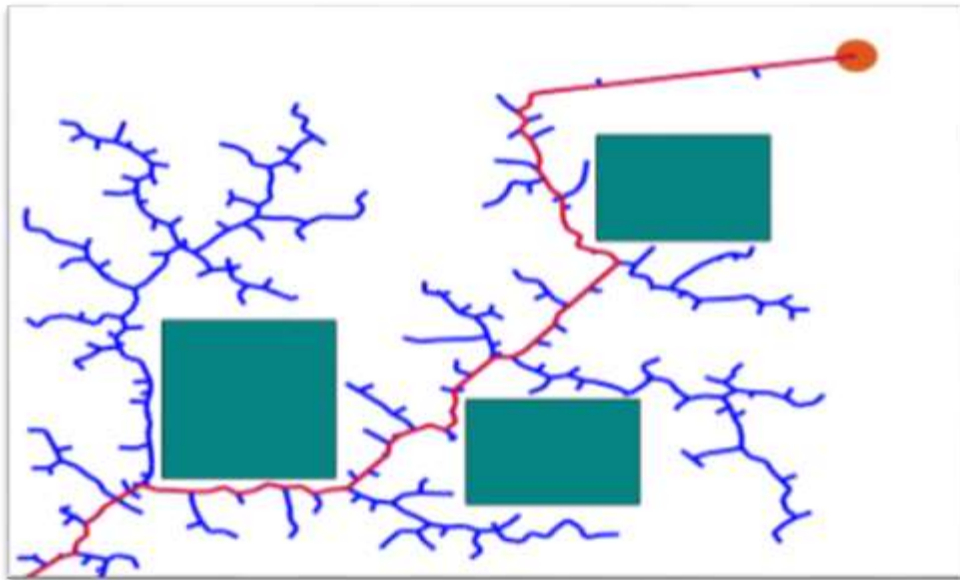


Figure 2.4: RRT* Structure [39].

Generally, the PRM probabilistic roadmap method of path finding in motion planning relies on graph-based algorithms [104]. It generates probabilistic roadmaps and checks for the most

available paths. First, a random sampling of several points from the map is done. All those that lie on an obstacle are discarded. It then connects any two points if there is no obstruction between the points. A probabilistic roadmap is obtained [106]. Now, in the probabilistic roadmap, a proper search algorithm is implemented in order to find the best path. Algorithm PRM is probabilistically complete. This means that if there exists a feasible path in the roadmap, this algorithm will eventually find it. PRM is a methodology of path planning using graphs to find pathways. In essence, it generates probabilistic roadmaps and then computes the best paths based on these probabilistic maps [107]. It starts by randomly sampling a set of points on a map. Points falling on obstacles are discarded; connections are made between adjacent points which are not obstructed. A probabilistic roadmap thus results. Hence, the probabilistic roadmap becomes the arena in which to apply a suitable search algorithm in order to determine the optimal path. The algorithm PRM is probabilistically complete, since, if a feasible path exists in the roadmap, the algorithm will eventually find it.

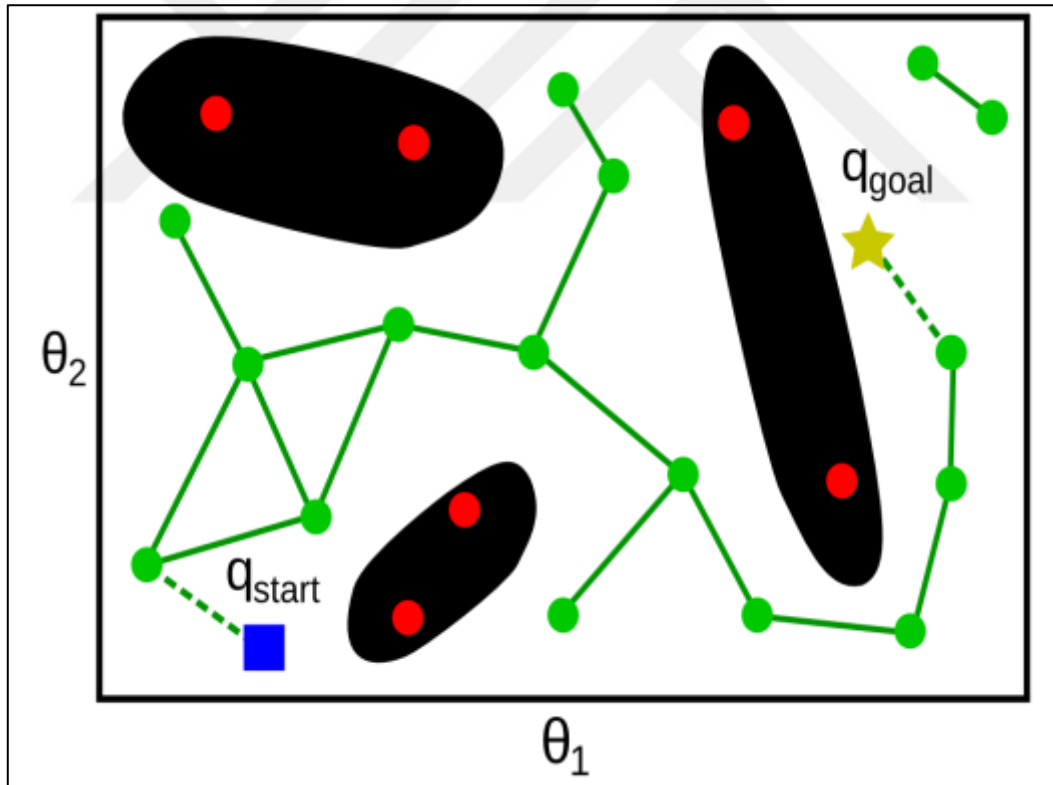


Figure 2.5: PRM Algorithm Structure [42].

2.3 OUR PUBLISHED PAPERS RELATED TO THE GRAPH SEARCH STRUCTURE

2.3.1 Nonholonomic Path Planning for A Mobile Robot Based on Voronoi And Q-Learning Algorithm

This paper addresses a differential route planning problem using Voronoi and Q-learning algorithms for the mobile robot. The challenge of finding feasible routes in static environments with obstacles must consider differential constraints that non-holonomic robots impose-for example, the inability of a car to move sideways or move backwards, having to go straight ahead with constant speed, and only left and right turns. Modifications are made to accommodate concave-shaped obstacles. Voronoi improves route planning by enhancing the Q-learning process using an enhanced Q table to reduce the space and time complexities and depends upon the prior knowledge brought by the agents. In simulation, it was in a 2D environment; the map style varied. A comparison was made to other methods hence giving efficient trajectories with improved performance.

Some non-heuristic methods adopted in path planning were discussed in this paper, including cell decomposition, Voronoi diagram, and B-Spline curve. Cell decomposition is two-pronged: exact and approximate methods of dividing the search space into simple cells and defining the relations between the cells. The exact one detects obstacles in the workspace and builds cells to determine free space; however, it is rather problematic for high-dimensional environments. An approximate approach was thus put forth. B-splines are mathematical functions that can facilitate curve generation based on control points rather than taking all the points into view. One modification was made by adding extra control points near obstacles and shifting these into clear areas. Methods are also used where control points can be adjusted according to the threshold set by the user, for example, altitude in enemy terrain when generating the paths of UAVs. The Voronoi diagram is applicative in a wide array of areas within the field of computational geometry. Its broad ambit has covered many applications in computer graphics, especially with regard to collision detection and motion planning. Normally, a Voronoi diagram is based on Euclidean distance, which refers to the method by which a 2D plane is divided into regions. A Voronoi diagram forms these regions from a set of points where each point p_i .

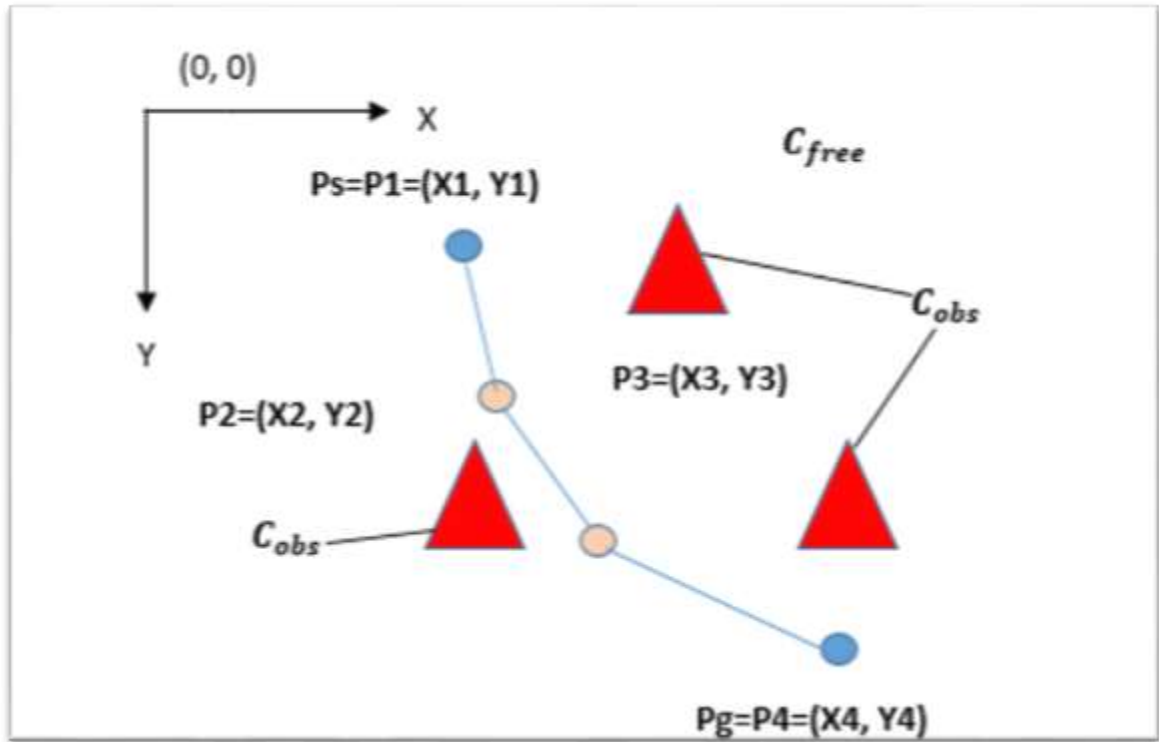


Figure 2.6: Free Path Inside the Environment Planning.

Q-learning is based on the reward and punishment mechanism; when a good result comes out, it gets a reward, while poor results keep punishments. The parameter r defines the conditions and actions along with other learning parameters denoted as a denotes the importance of a state-action pair for a given time. The algorithm stores such approximate values for all possible state-action pairs; hence, in every given state, the algorithm would make decisions based on learned experiences. They present the use of Voronoi diagrams and Q-learning for non-holonomic path planning with the aim of obtaining a collision-free path. This is to identify the optimal path of a mobile robot and make use of the advantages offered by Voronoi diagrams for accelerating the searching process. A robot should move from its source position to its target by choosing the best path. The environment is detected, the obstacles are located using the properties of Voronoi at the start of the experiment, and then the robot is guided to its final goal by Q-learning. During the simulation, several map types have been used in which the start state is represented by the green point, the destination is represented by the blue point, and the red triangle marks the concave-shaped obstacle.

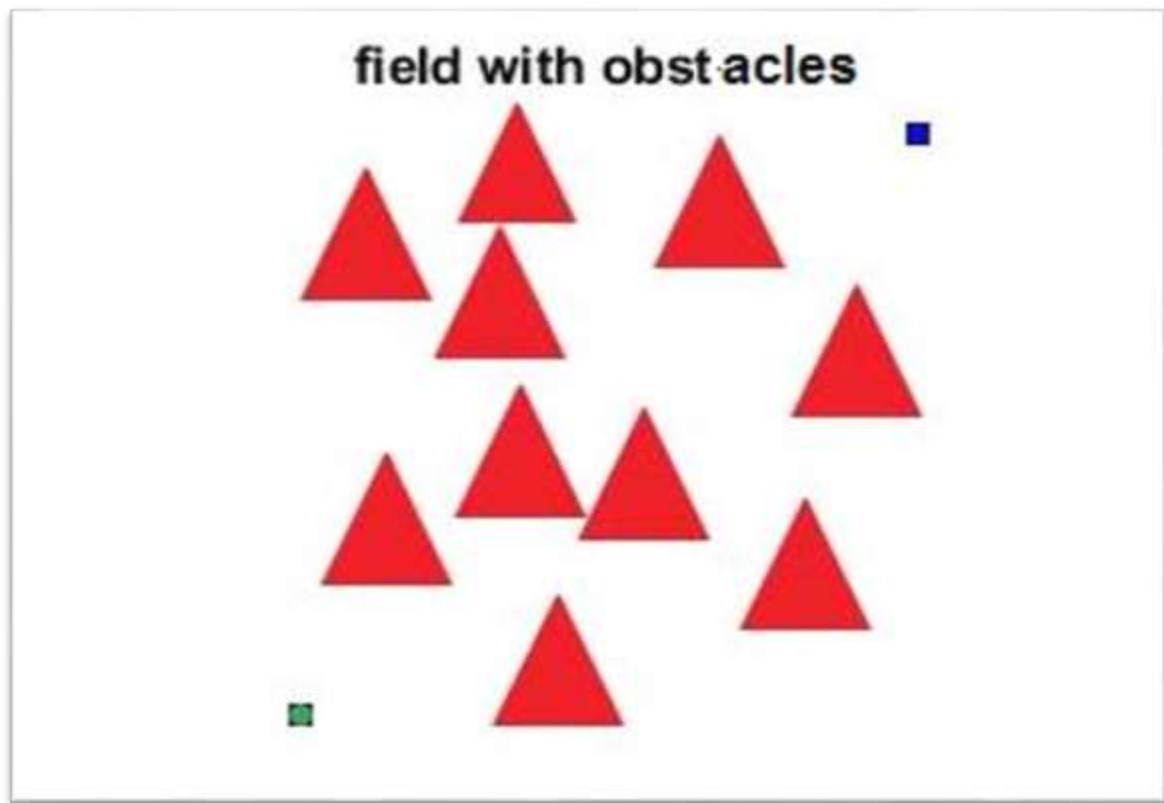


Figure 2.7: The Under-Test Environment.

That is, Voronoi diagrams initialize the environment and map out routes that keep the robot furthest from the obstacles. Once all the possible paths are generated, Q-learning takes over and looks upon each state by taking actions, receiving rewards until the transition to the next state reaches its goal. The last but not least step will be to build the optimal path from start to goal. This is done in a static environment with arbitrarily positioned obstacles and having arbitrary start and end positions. The research targets an improvement in stationary settings by enhancing learning and skill development. The simulated results show improved obstacle avoidance, improved time performance, and effective space usage for an improved trajectory through Voronoi diagrams.

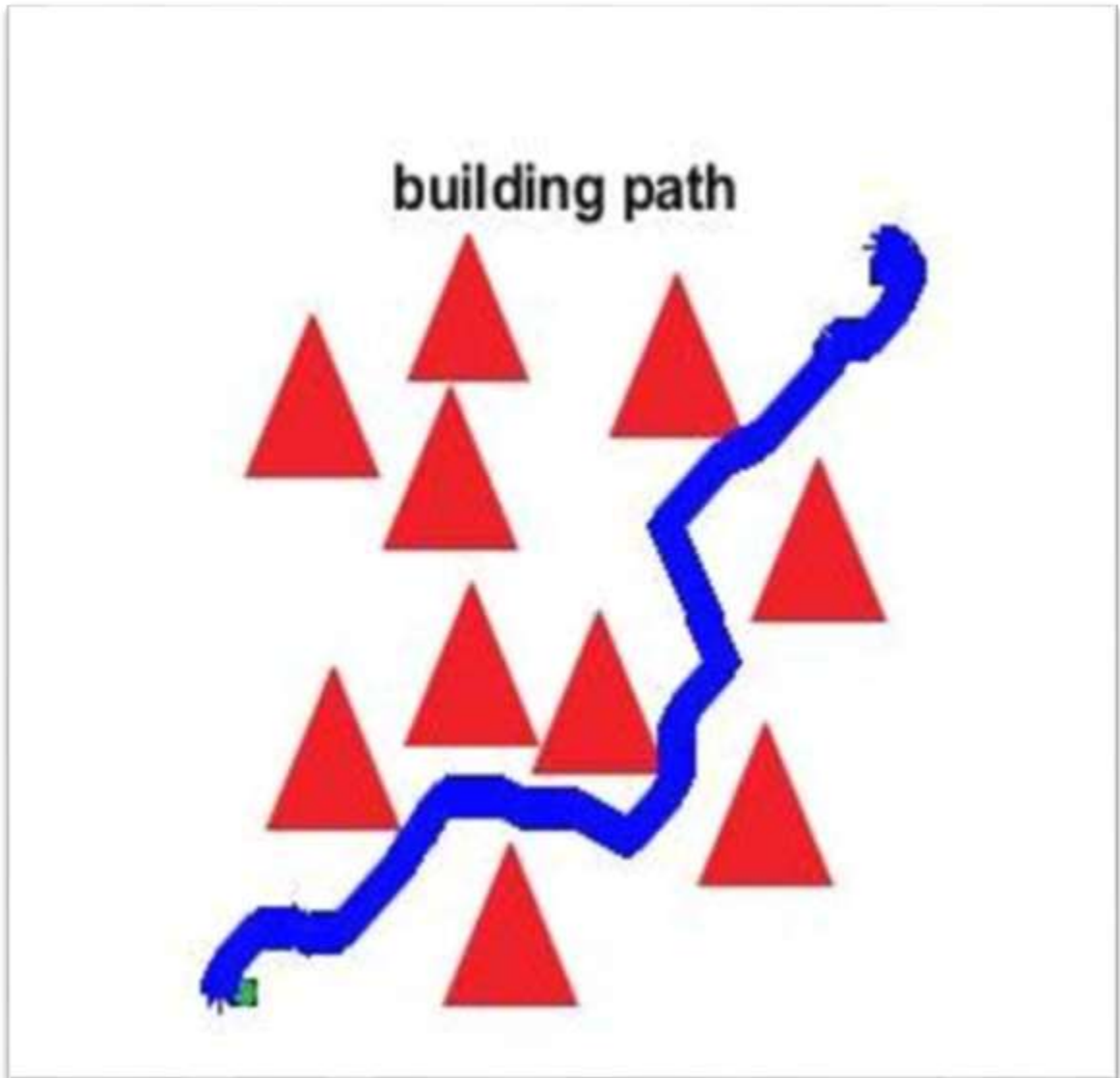


Figure 2.8: The Result Obtained by The Proposed Work.

2.3.2 Car-Like Robot Path Planning Based on Voronoi and Q-Learning Algorithms

This paper addresses the solution of the differential path planning problem of a mobile robot using Voronoi diagrams and Q-learning algorithms. The main emphasis has been given to finding the optimal route between any given starting point and target in a dynamic environment with obstacles. Considering that the robot is a car-like vehicle, some important modifications have taken into account, such as the change of orientation due to the differential constraints. It is assumed that the movements of the robot are restricted only to moving forward at fixed speed and making left and right turns. Voronoi diagrams improve the process of path planning through mapping of the environment, hence improving the

reliability of the planning and reducing the complexity associated with Q-learning algorithms. Voronoi diagrams help in the management of space and time complexity in relation to pathfinding by updating the Q-table based on the new state. The method is evaluated in a 2D space, and the results are presented as improvements with respect to time, speed, and length of path. It also proved adaptive to various map styles.

A robot, in different scenarios, has various alternatives to reach a defined target; however, it selects the best choice depending on factors such as the shortest distance and minimal time consumed. Of these, the most often adopted criterion is the minimized distance. Path planning concerns finding the shortest possible path while satisfying constraints such as collision avoidance in an open environment, as considered in [85]. Non-heuristic established approaches to path planning include unit decomposition, Voronoi diagrams, and B-spline curves. Unit decomposition involves two methods that are in common practice, namely, exact and approximate cell decomposition. The exact method decomposes the search space into simple cells and finds their adjacency, whereby obstacles and construction of cells, according to [105] and [108], are clearly identified. Although this method represents free space accurately, it is hard to find the exact free space in high-dimensional environments; hence, approximate methods have been adopted. For refinement in path planning, extra control points, which are closer to obstacles, are used in B-spline curves. A method can be devised that translates such control points from obstacles to free space, as mentioned in. Threshold heights can also be defined by the users to generate paths for the drones, allowing drones to navigate through even more complex spaces.

In paper [96], the Voronoi D* Algorithm was proposed to further improve route planning performance and meet the distance specifications for mobile robots. Yet, this algorithm may suffer from local minima problems. Similarly, approaches for complex obstacle environments mentioned in [97] also have challenges on trajectory quality. Works, while few of them focus on route improvement rather than route development, use ML algorithms and especially QL for path planning in unknown environments. Q-learning algorithms are preferred due to their simplicity and convergence properties; hence, they are among the most employed RL algorithms [100], [102]. On the other hand, nonholonomic path-planning solutions were provided by using the RRT algorithm, well known for poor performance of the obtained trajectory. It is designed to handle runtime changes in the environment

elegantly. This is realized by the updated Q-learning approach where it does the change in a Q-table according to the conditions. This method uses the knowledge of Voronoi diagrams to reselect the shortest path whenever changes come into view.

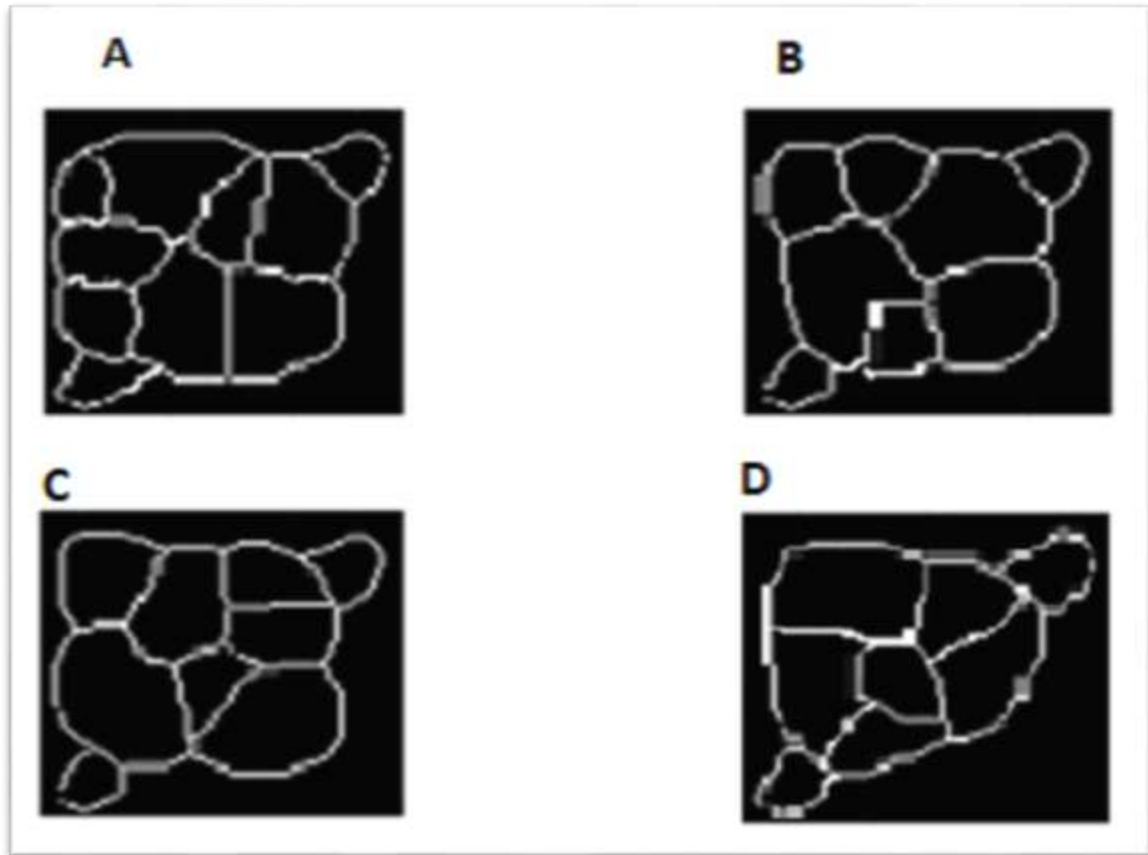


Figure 2.9: Voronoi Diagram Environment Result.

The environment is changed quite frequently, requiring the mobile robot to be able to continuously avoid obstacles on its path to the goal state. This work is applied to a dynamic environment that has obstacles positioned at random and with randomly assigned start and end positions. As discussed in [109], the following improves path planning both in a static and dynamic environment. We also include a speed profile menu that depicts variations in the speed of the robot as it navigates through obstacles.

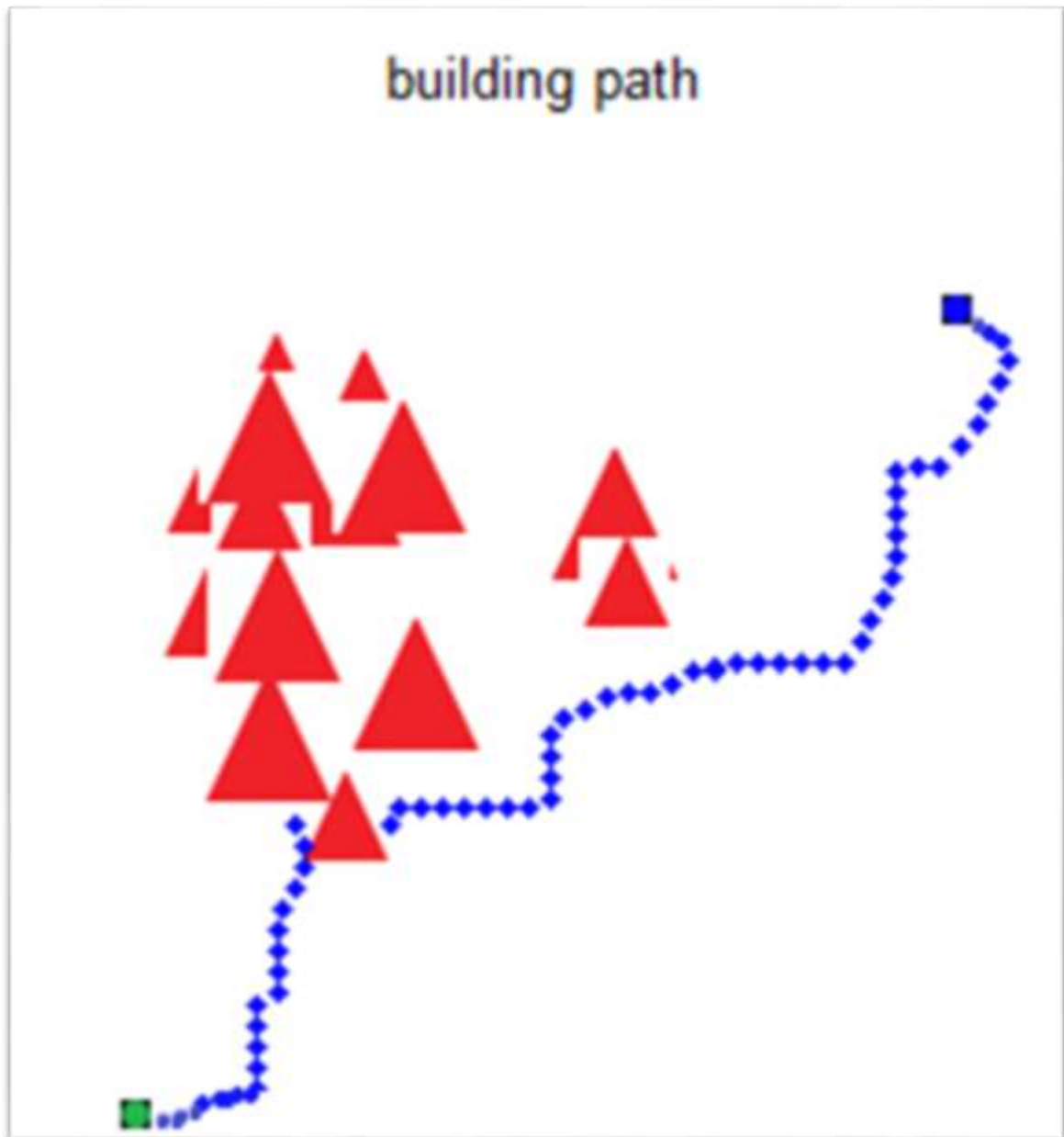


Figure 2.10: Result Obtained by Proposed Method.

The simulation results show that the application of Voronoi diagrams expresses significant enhancement in obstacle avoidance capability and improves performance both temporally and spatially. In contrast to some strategies, such as RRT, which often gives suboptimal trajectories, this strategy computationally gives more accurate trajectory planning. This VD approach helps avoid such issues and provides more accurate path planning. In contrast, D* algorithm certainly performs well in finding the goal. However, it may be trapped in local minima, and it may lose time. This is where learning algorithms have attained high marks,

and several implementations also outlined that exact time estimation and accurate learning metrics are quite relevant. Based on this view, our approach will exploit VD to determine early on whether reaching the goal is possible, which then saves the system from losing time on infeasible paths. By integrating VD with QL, we improve the learning process with prior knowledge about the environment and result in improved rewards over penalties. The integration also supports refining the reward function in QL and adapts it to changes in the environment for optimization of overall performance.

2.4 MULTI-AGENT PATHFINDING

Multi-Agent Pathfinding, or MAPF, is a broad and well-explored area in the studies of computer science and robotics that focuses on how several agents can coordinate their motions in some shared environment without collisions. This coordination challenge has wide applications in logistics, robotics, and autonomous vehicle management. To perform the problem of MAPF, research has created different methodologies, starting with classical search algorithms like A* and Dijkstra's, adapted to take into account multiple agents. These early solutions enabled individual agents to make independent plans of their paths but faced severe limitations once the density of agents increased, resulting in frequent path conflicts [110]. One of the most important milestones in MAPF research was arguably the development of the Constraint-Based Search framework: one designed to efficiently handle these conflicts. It splits the search into two levels: a high-level search for performing conflict detection and resolution, and a low-level search for doing individual agent pathfinding. This is an optimal approach, as it introduces specific constraints whenever a conflict is detected to guide each agent toward a collision-free path.

Besides, there were priority-based and cooperative pathfinding methods developed for its further efficiency. Priority-based pathfinding gives hierarchy among agents. While higher priority agents move along their paths, lower priority agents change course as needed to prevent collisions. This can, of course, result in faster computations, though at a possible cost of less-than optimal paths for those agents of lower priority. In contrast, cooperative pathfinding allows agents to communicate and make reservations about their paths into the near future, balancing efficiency with a more collaborative process of pathfinding [111]. Both methods form a very effective strategy wherein real-time coordination and quick decision-making are involved.

As MAPF scenarios became increasingly complex, researchers began to work on decentralized and distributed methods. While agents in these frameworks calculate their own paths, they interact to avoid collisions[112]. The distributed methods have included such as Token Passing and Decentralized Conflict-Based Search; these have considerably benefited large-scale environments featuring many agents for which the computation may be infeasible by a centralized approach. Decentralized methods hold especial relevance with respect to such restricted-communication infrastructure scenarios where localized decision-making processes can remain dynamic and change by ever-changing environments. Event works on MAPF involve reinforcement learning and lifelong MAPF, where the agents continuously adapt and replan routes at any time. The RL-based MAPF enables the agents to learn policies of navigation and conflict avoidance in dynamic and uncertain environments, such as a warehouse with moving obstacles. On the other hand, lifelong MAPF deals with recurring or continuous tasks where agents are able to plan and replan routes efficiently since they perform continuous missions [113]. These methods are mainly useful in complex and dynamic environments where agents need adaptation to new obstacles and tasks.

Heuristic optimization has also played a very important role in MAPF to get faster and more practical solutions. Besides that, algorithms like A* that treat cooperative constraints efficiently-offer flexible choices toward generating high-quality paths without rigidly enforcing optimality when slight sub-optimality is tolerable in a static environment as mentioned in [114]. In real-world applications, MAPF solutions are being deployed onto real problems including robotic coordination in automated warehouses, air traffic management, and even swarm robotics. While much has been accomplished, several challenges remain open, including scalability, adaptability to dynamic environments, and integrating MAPF with machine learning and other AI techniques that enable enhanced autonomy. Community research increasingly addresses these challenges and positions MAPF for application in real-world settings by considering hybrid solutions that integrate traditional algorithms with modern machine learning methods. The development of MAPF continues with classical pathfinding, constraint-based optimization, decentralized planning, and learning-based methods. Theoretical underpinnings and practical advancements at the forefront of MAPF will surely solve increasingly complex and realistic environments taken from many different

industries. With these applications comes the increasing need for hybrid and adaptive methodologies driving future innovations to scale MAPF toward larger and even more dynamic multi-faceted environments. This system allows agents to apply only four kinds of movements-up, down, left, and right-at a constant speed. These actions link two states by an edge. Moreover, agents can apply a no-operation action, say staying in the same location. Time is discretized: every action corresponds to one time step, each with some cost associated. When the standard method of MAPF finds a conflict-free path for each agent to reach its target state, the solution for each agent. It is stored as a sequence. It comprises a sequence of actions to be taken by an agent for the agent to reach its destination at the starting state [115]. Conventional multi-agent path-planning algorithms are usually confronted with high energy consumptions during the execution of cooperative tasks. Most of them feature overlaps along the paths due to the single-agent-single-task configuration. In order to solve these problems, "many-to-one" cooperative planning was developed. This approach has also come to face some challenges in having a large meeting point search space and selection of impractical locations where a task handover can take place [24]. The major problem that conventional MAPF problems have always faced is the search for an optimal solution that minimizes costs with conflict avoidance for all agents to their destinations efficiently. It deals with a certain kind of agent, namely omnidirectional robots with three motors. Such robots are different from any other robots as, without the added complexity introduced by the movement constraints, they can take on, or even possess, different directions and orientations. In [28] they addressed the problem of Multi-Robot Path Planning in continuous space for finding conflict-free paths. Two factors are major contributors to the problem's complexity. First, decisions were made combinatorically due to the presence of multiple robots, exponentially increasing the search space of the problem. The second is the continuous space that introduces potentially infinite states and actions. they handle this using a two-layer approach, whose lower layer is based on a sampling-based planner, namely the Safe Interval RRT* (SI-RRT*). The main difficulties for this lie not within the robot's capability but rather within the constraints of the system. These are further dug in the nature of the environment and the capability of the algorithms in manipulating the immediate surroundings of varied settings. In other words, as much as omnidirectional robots provide flexible movement, effectiveness in operation is determined by how the algorithms are able to adapt and handle dynamic conditions of their surroundings [44]. This means considering

environmental factors and ensuring that algorithms can respond well to changes and obstacles on the ground in real time.

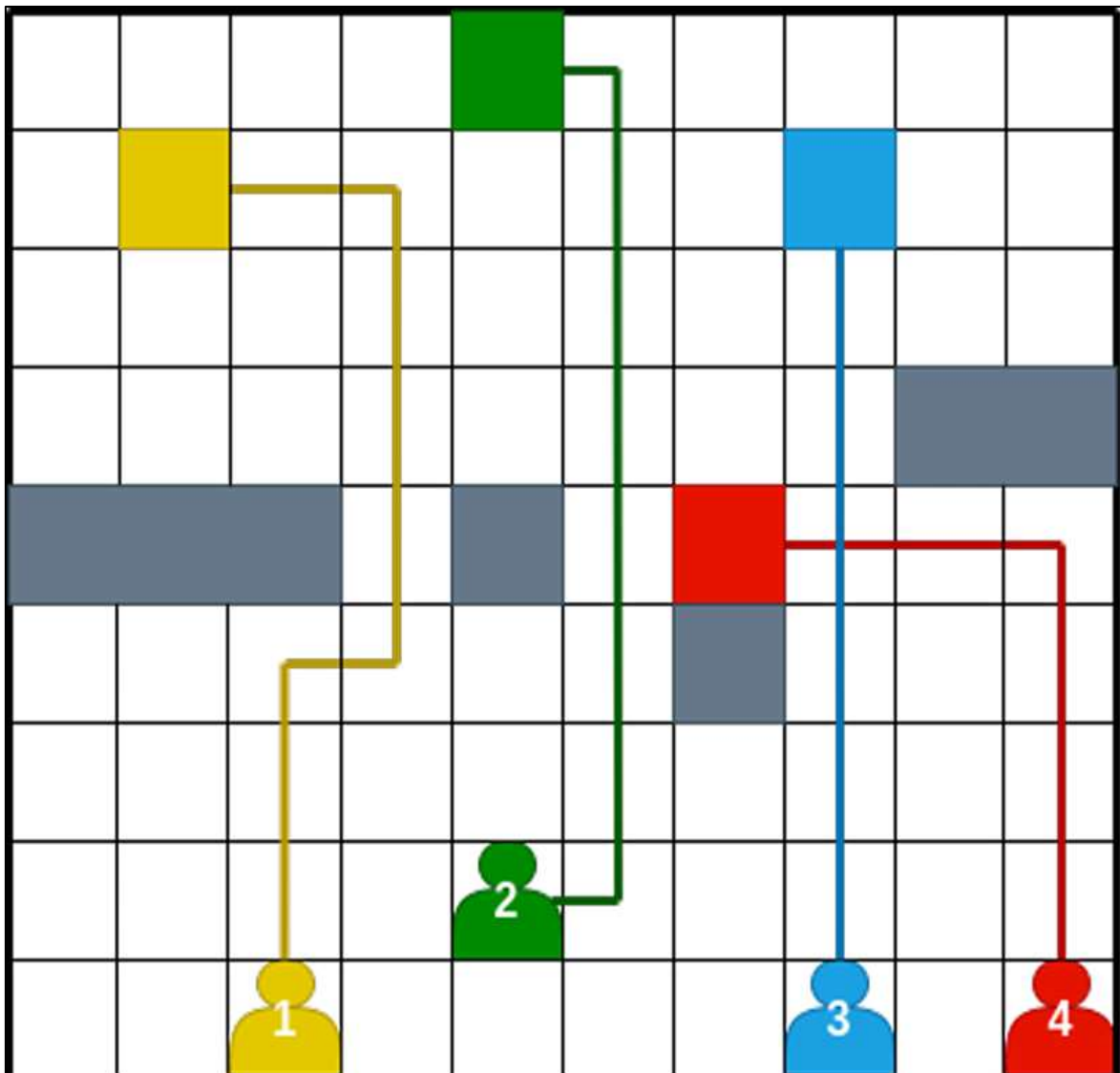


Figure 2.11: Shows An Example of MAPF Inside the Environment.

The example showed there are four moving agents trying to reach their targets. In this case their paths must be planned without collision.[1]

2.5 COLLISION FREE PATH WITH PICK AND DELIVERY OPERATION

In the case of MAPF with pick-and-delivery, a collision-free path for agents that have to move from a source location to pick up an item and then deliver it to a specific delivery location is to be computed. This further complication of handling the pick-and-delivery tasks

involves not only avoiding obstacles and other agents but also coordinating locations to pick up and drop off items. This challenge is accentuated by the dynamic nature of the tasks and dependencies that are created between agents, whereby some have to wait until others perform certain tasks before theirs can be pursued, hence complicating the collision-free path planning challenge [116]. The standard procedure for solving a MAPF problem involving pick-and-delivery operations normally involves an environmental modelling phase, whereby the environment is represented as a graph where locations are represented by nodes and the edges represent the available paths. Each agent is to navigate to a pick-up location, collect an item, and then move it to the drop-off location.

The main approaches toward collision-free paths have to do with task assignments and prioritizations. Task assignment would involve assigning each agent to a specific task in a manner such that the trajectory of agents across different tasks is least overlapping. In this case, priority assignment will help avoid conflicts wherein an agent has tasks that are related to critical timings or dependencies. Techniques like Cooperative Pathfinding or CBS allow dynamic adjustments in the paths taken by agents such that they may reroute and avoid areas where other agents presently are or are en route. Also important are the synchronization strategies when several agents are to use the same locations for pick-up and delivery. These typically allow the agent to proceed only when the other agent has gone past the shared location. A priority inheritance mechanism may be useful in creating a temporary increase in an agent's priority when, given its current path, this would otherwise cause a bottleneck or conflict with others. Finally, optimization techniques balance the timing and efficiency of each route by every agent in such a way to minimize both the distance travelled and the number of conflicts occurring. Advanced heuristics and scheduling with time windows can optimize these routes in such a way as to make agents arrive at pick-up or drop-off locations without delay, contributing to fluent and collision-free motion. The combination of pathfinding algorithms with synchronization and optimization allows MAPF to incorporate pick-and-delivery operations in such a way that solutions to extra challenges caused by dependencies between tasks are enabled, but in a way that allows efficient multi-agent logistics in complex structured environments, including warehouses, transportation networks, and robotics applications.

The most promising alternatives will probably involve somehow decoupling the problem at hand, rather than exploring the entire shared space. It works because it solves only the coupled parts of the problem as necessary. A strategy that has gained much attention lately is called Conflict-Based Search or CBS. It precisely targets collisions arising in the problem and solves them. Because it is just targeting conflicts, CBS can take much better care of joint-space search complexity, and its overall performance increases dramatically. These two factors bound the scalability of joint-space search: it requires a very large search space, and there is strong coupling between agents. Due to this fact, other ways of trying to make joint-space search more efficient are being researched. CBS proposed a search-based scheme for solving the MAPF problem on a constraint tree. Indeed, several implementation factors can make CBS significantly performant: Conflict Selection-choice of which conflicts to resolve is crucial for the actual optimization of the search to perform much better. Heuristic Guidance-whenever applying good heuristic guidance allows the search in a promising direction that reduces the overall search time. The stronger the imposed constraints are, the more effective the conflicts can be resolved to improve the performance of the algorithm. By paying attention to these factors, CBS can be fine-tuned in handling MAPF problems for better results and quicker solutions.

The lifetime MAPF problem addresses the problems of how to assign the jobs to multiple agents and who performs each task. These questions are at the core of many real-world systems' ability to function effectively. The problem of job distribution among agents has widely been researched along the lines of the Vehicle Routing Problem (VRP) [74]. However, in VRP, the centre of gravity is more towards an optimal solution with respect to the cost of the path without taking into consideration the issue of potential collisions among the agents. It also relies a lot on data in road networks and does not take into consideration dynamics due to agent interactions. On the contrary, collision avoidance among agents plays a great important role in applications involving automated warehousing and computer games. [65] introduced the so-called Multi-Agent Pickup and Delivery, or MAPD, problem to incorporate this into the mainframe. This generalizes the VRP into consideration of possible collisions between agents with the tasks they have to perform. In the light of these considerations, MAPD will allow for a more complete framework to handle interactions in multi-agent environments, so that operations can be both safe and efficient.

2.6 LITERATURE REVIEW

The history of planning algorithms encompasses various methods, each describing different heuristic and evolutionary and hybrid approaches towards route planning challenges. MR is used to deal with tasks that are complex in nature. A single robot may perform many tasks proficiently, but a multi-robot system can perform different activities simultaneously. However, multiple robots bring their own challenges while being managed. Such diversified applications comprise dynamic mission planning [2], collaborative construction, multitasking assignments, and environmental mapping. Each robot in the system is assigned a particular mission, which makes the coordination of multiple robots in uncertain environments dynamic and challenging. Besides the structural complexities of the environment, every robot also creates dynamic obstacles affecting the performance of other robots. Multi-agent reinforcement learning is an important area of reinforcement learning, with the main focus being on sequential decision-making problems that envelop multiple agents. In MARL, the agents share actions and knowledge in order to reach goals that are either common or individual.

The complexity increases when applying MARL to such problems as coordination of a group of mobile robots navigating across different locations. The problem that most closely relates to this situation is the traveling salesman problem in its generalization of having multiple depots: Multi-Depot Traveling Salesman Problem MTSP [117-120]. Many of these issues have various approaches to finding a solution. For example, [14] proposes a dispute resolution technique in which agents manipulate their speeds and movements to avoid disputes. This is effective in dynamic conditions where the actions of the agents must be adjusted continuously. Also, an improved algorithm by [121] limits the needs for re-planning since the adaptation towards new information or changes in the environment is much simpler than before. Moreover, the authors proposed a parallelized approach for MAPP in. Indeed, parallel computing is adopted for path planning problems in a more efficient way. Through experimental comparisons, one can find that the parallelized approach will perform better than other conventional methods in a static scenario. Its effectiveness arises from how to optimize the mutual paths of multi-agents. These are hence the ongoing developments in the domain of MARL system efficiency and effectiveness enhancement in complex settings where coordination and adaptive decision making becomes paramount.

In [17], the route planning of conflict-free path planning in automated container terminals was done using MARL for AGVs with an amalgamation of kinematic constraints and omnidirectional motion capabilities, and it was quite challenging. A* is one of the general pathfinding algorithms commonly used. In any case, A* does not work well in MAPF. It requires extremely accurate environmental models, and even then, does not function well in very complex conditions [23]. It is possible to use A* in order to simulate particular scenarios; this test, however, shows that the method has some particular weaknesses, meaning more sophisticated methods will be required [19]. Future research will be directed at the experimentation with advanced techniques in highly complex environments. These works aim at enhancing performance while reducing complication by including advanced strategies. In this regard, one of the recent works is introducing the MAPF-POST method in. This work considers differential drive robots and deals with maintaining safe distances between the robots with their velocity limits. It provides a generalized version of CBS for large agents which occupy multiple grid cells. MAPF-POST has especially been fitted for pickup and delivery scenarios and combines token-passing techniques with SIPP. Since these methods are combined, the approach can plan the paths more efficiently and reliably in dynamic and crowded environments.

It also extends to a roadmap-based planner that takes into account movement speeds and a grid-based planner created for any-angle motions, using a variant of Speed-Indexed Path Planning, or SIPP [45]. This marriage gives the capability to handle many different movement capabilities and enhances pathfinding precision. The other key category of the research in distributed collision avoidance is the non-holonomic robots that cannot move in arbitrary directions without changing orientation [46]. These have been classically controlled by methods such as artificial potential fields, dynamic window approaches, and model predictive control. The Reciprocal Velocity Obstacle algorithm is decentralized to avoid collisions while having possible effective communication [47]. This allows the method for the robots to navigate safely among shared spaces. However, many of these techniques have much challenged complex or stringent environmental constraints. Among the most frequently encountered problems is the need for rapid replanning based on a multitude of short-term changes, which complicates the implementation process [48]. In this article, attention will be turned precisely to warehouse and industrial applications whereby many robots are needed for most optimal labour-intensive tasks [112]. Indeed, such an

environment does require very efficient re-planning in order to cope with the constantly changing conditions of the environment for sustaining operation efficiency [123].

Table 2.1: Illustrate Related Surveys on The Avoidance of Obstacles in A Different Area.

Study	Kind of method	Environment setting
[24]	Soft Actor-Critic	Partially observable environments
[25]	DWA	unknown environments
[26]	IQL	Warehouse
[27]	PSO	Static and dynamic En
[28]	Heuristic	Normal map-based WH
[29]	RNN	Warehouse
[30]	BFS	Warehouse
[31]	CBS	Game theory
[32]	A*	Warehouse
[33]	DEEP	MIX environment
[34]	Antennae Search Algorithm	Static environment
[12]	RRT	Static and dynamic environment
[37]	RRT*	Dynamic environment
[38]	DTW	Static and dynamic environment
[39]	RNN	Static and environment
[40]	Q-learning	Mixed
[41]	Voronoi	Static and environment
[42]	ACO	Unknown environment
[43]	TS	Unknown environment
[45]	NN	Dynamic environment
[46]	Fuzy	Dynamic and static environment
[47]	AGT	Mixed
[48]	GN	Static and dynamic
[49]	ICS	Dynamic environment
[50]	CT	Dynamic environment
[51]	M*	Unknown environment
[52]	Dijkstra	mixed

2.7 SUMMARY

We highlighted the main challenges in MAPF and discussed typical techniques with their limitations. Pathfinding solutions can generally be categorized into three types, including those for MAPF: Optimal Approaches: These methods guarantee an optimal solution. These approaches are usually computationally expensive and require a lot of resources to execute, generally speaking, if considering complex scenarios. Bounded-suboptimal approaches are methods for finding good solutions within a certain bound of optimality. In other words, it is a trade-off between solution quality and computational efficiency to provide reasonably good solutions without the high computational cost of optimal approaches. Unbounded-suboptimal approaches are those oriented toward speed and scalability, not guaranteeing that

a solution will be returned within a certain bound of optimality. The focus here is to come up with a satisfactory solution in as short a time as possible rather than one that is near the best possible. Each category has its strengths and limitations depending on the problem requirements and constraints.



3. METHODOLOGY

The proposed structure is identified in this section, as well as an explanation of the main aim of MAPF which is provide the agents with the capability to reach their goal positions from their starting positions without crashing with each other or any obstacle present. As agents increase, the problem also grows and gets complex; hence, finding good algorithms and methodologies for the sake of optimal pathfinding solutions is at hand. First of all, the MAPF approach requires a proper problem formalization, which is typically modelled as a graph whose nodes represent feasible positions, while edges denote possible movements between consecutive positions. This kind of graph is complemented by additional information: starting and goal positions of each agent, obstacles, and other movement constraints. The problem can also be framed as an optimization task by seeking to minimize cumulative costs, being some metrics of travel time, distance, or energy consumption, with the guarantee of collision-free motion for each agent. Since several agents are traversing around the environment, there is likely a possibility that their paths might cross. Thus, the method involves conflict resolution techniques to detect and mitigate such conflicts. Other techniques, such as priority-based pathfinding methods, simply give each of the agents a priority level so that the higher-priority ones keep their paths, while the rest are rerouted. Besides, it is also possible to apply flow-based optimization techniques in order to find those global movement patterns that minimize conflicts among agents, hence offering even smoother navigation. In addition, the correlation and its role in calculating subcarrier characteristics were explained.

3.1 MULTI-AGENT PATHFINDING

The Multi-Agent Pathfinding problem is a well-known abstract model for the collaborative path-planning problem. The MAPF problem is defined on a graph structure $G = (V, E)$, where V denotes a set of vertices and E characterizes the edges connecting these vertices. In this work, we restrict the graph to be an undirected simple graph [123-135]. Undirected Graphs: The edges have no direction, meaning it can move from one vertex to another connected vertex in any direction. Simple Graphs: For every pair of vertices, there is at most one edge between them; second, there are no loops, meaning no vertex has been connected to itself by an edge. MAPF Problem: A team of agents K navigating along the graph G . G hissed

simultaneously. Each agent needed to make a plan for its route to avoid collision with other agents while moving from its start position to the goal. This, therefore, acts as a structured framework towards addressing challenges in multi-agent coordination and path finding within a shared environment [136]. In the problem of MAPF, pathways are constrained such that the location that an agent can occupy at a particular time step is unique. This rule prohibits agents from appearing in more than one location in space at any one time or disappearing from the graph [137-146]. This rule creates a problem when there is a missing time step in the planning of the pathways. In particular, in cases when the trajectory did not consider all time steps that are needed to reach a goal, collisions with other agents can still occur even if a goal has already been reached. This aspect is important in MAPF, as it shows that for path planning, completeness and optimality in avoiding collisions, keeping agents out of the way of each other-even while their approach to the destination-cannot be compromised.

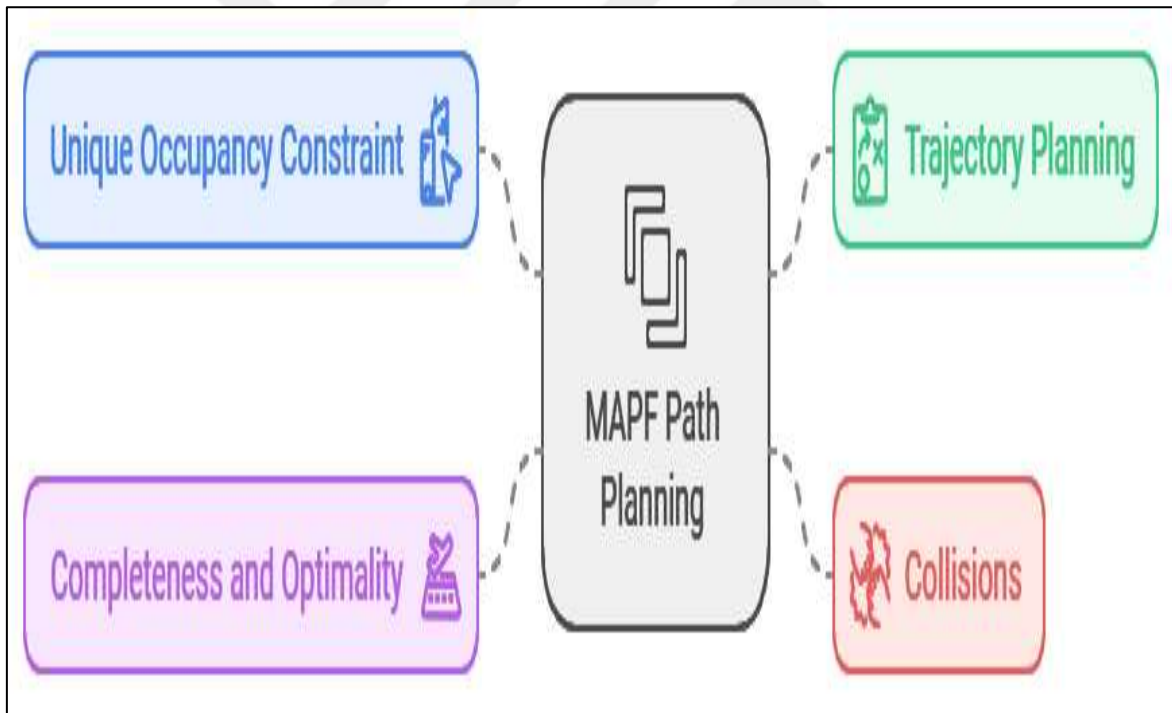


Figure 3.1: MAPF Map Overview.

3.2 COST METRICS

The SOC metric quantifies the total cost of a solution represented by the sum of the costs of its constituent pathways. While SOC represents a cumulative measure of costs, it does allow arbitrary assessment of particular pathways, and this can make interpretation complicated. In the present study, we employ a simpler statistic, referred to as time spent off target. The metric describes how many times steps an agent spends away from the goal node to which it is assigned. It simplifies the evaluation process because it allows focusing on how deviations from target directly affect the cost—more precisely, the possible cost impacts of every deviation along a path are limited. For example, any deviation can add up to two units to the total cost—which encompasses the penalty for moving away from the goal. In that respect, this will ensure that the representation is well-defined and workable by the model in order to measure the implication of pathway changes on the overall solution.

To construct ordering mechanism that exhibits the above-mentioned properties, we consider node transitions as an implicit alphabet. Since there are at most a finite number of nodes in a graph, we can represent such an ordering mechanism with a number in base ten. For a given route we construct a decimal number representative of it. This number contains two parts: the integer component: describes the cost of the route. Fractional component: the fractional part is another major part of this encoding, which models further details around the route taken, including detouring from an optimal path. We are going to compare routes through their decimal representations to evaluate options. The agent shall always choose routes with small decimal values, for they mean the paths bring the agent closer to the goal on the other side, larger decimal values are meant to be avoided because they mean the paths increase the distance from the goal. This method of lexicographic ordering will help in the systematic selection of the best route options to pursue a goal. The cost node is the sum of the costs of all agents' paths:

$$\text{cost}(N) = \sum_{i=1}^n \text{cost}(\pi_i) \quad (3.1)$$

Where N is the current node, and n is the number of agents. To implement a priority scheme for the nodes, each is assigned an index in a range between 0 and a maximum number of possible indices. Such an indexing system would be implemented as follows: Priority Indices w.r.t. Objective-Directed Edges: The indices are first assigned to edges that lead directly

towards the objective. These are assigned in order of priority, with lower indices being given to the most favourable edges bringing the agent closer to its goal. Equal-Distance Edges and Pause: Once the indices for objective-directed edges have been set, indexing is assigned to the pause action. Afterwards, this indexing is extended to all the edges that lead to nodes falling into a position equidistant from the objective; hence options that are equally distant fall into this category. This is done by indexing the nodes that lie away from the goal; higher indices are attached to these nodes, and these carry a low priority index compared to their heading-towards-goal counterparts. This helps in an organized transition between the nodes and assists in decision-making to ensure one is moving closer towards the objective.

3.3 CONFLICT-BASED SEARCH (CBS)

CBS is one of the most representative algorithms to solve Multi-Agent Path Finding problems [167]. MAPF aims to seek collision-free paths for multiple agents, such as robots or vehicles moving on a shared grid or graph, from their initial positions to their target destinations while minimizing certain costs, such as time or distance. CBS works by decomposing the problem into two layers:

- a. Top level search: It manages constraints among agents to avoid conflicts.
- b. Low-level search: This calculates the individual optimal paths for each agent by considering given constraints.

A high-level search can be done on a CT, where each node represents a solution consisting of a set of paths and the conflicts among agents. The CT is a binary tree where each branching introduces a new constraint to avoid conflicts. The conflict can be defined between A_i and A_j as:

$$conflict(A_i, A_j) = pos(A_i, t) = pos(A_j, t) \quad (3.2)$$

Where $pos(A_i, t)$ refers to the position of agent A_i at T time. On the other hand, the algorithm splits, in case of a detected conflict, the current node into two child nodes by introducing constraints on the conflicting agents:

$$C(A_i) = \{(p, t)\} \quad (3.3)$$

$$C(A_j)=\{(p,t)\} \quad (3.4)$$

Each of these nodes will create sub-nodes to find solutions. At each node of the high-level search, the low-level search determines the optimal path that each agent can take given the current set of constraints.

If we have two agents A_1 and A_2 if they conflict at time $t=5$ at position $p= (2,3)$ the constraints will be as shown below:

For A_1 :

$$C(A_1) = \{((2,3),5)\} \quad (3.5)$$

For A_2 :

$$C(A_2)=\{((2,3),5)\} \quad (3.6)$$

In MAPF, priority is a mechanism employed to resolve conflicts between agents, where some have precedence over others. Instead of branching on both in the case of detecting a conflict, a priority-based approach can continue along the path from one agent, while the other agent has to make an adjustment in their path in order to avoid conflict. The use of priorities greatly simplifies conflict resolution, especially in a real-time or dynamic environment, which may come up with solutions faster. It is based on the principle that higher-priority agents shall be less restricted, while lower-priority agents must replan their paths to accommodate these higher-priority ones.

When two agents A_i and A_j have a conflict at position and time priority will be as:

- a. Agent priority hierarchy: For example, agent A_i has a higher priority than agent A_j at a specific point.
- b. Dynamic priority assignment: Priorities will be determined depending on the distance to the goal or the nearest node.

Consider the priority of an agent A_i if the priority of $(A_i) < \text{priority}(A_j)$ the agent A_i has a higher priority and keeps its current path. While the agent A_j with the lower priority should then re-plan its path to avoid the conflict, and the re-planning will be as:

$$C(A_j)=\{(p,t)\} \quad (3.7)$$

CBS begins by designing the shortest pathways for each agent individually, which can be completed quite rapidly. These pathways were initially meant to avoid collisions. Conflicts are detected and resolved to get a collision-free solution. If a collision occurs between two nodes (for example, nodes a and b), CBS recursively examines two options: one with a limitation prohibiting node a from being in a cell x at time step t, and another with a constraint preventing node b from being in a cell x at time step t. At the top of the binary constraint tree, each node N holds a collection of constraints. Constraints can be either a negative vertex constraint (a,x,t_i) that prohibits node a from existing in cell x at time t_i or a negative edge constraint (a,x,y,t_i) that prevents node a from migrating between cells x and y at time t_i.

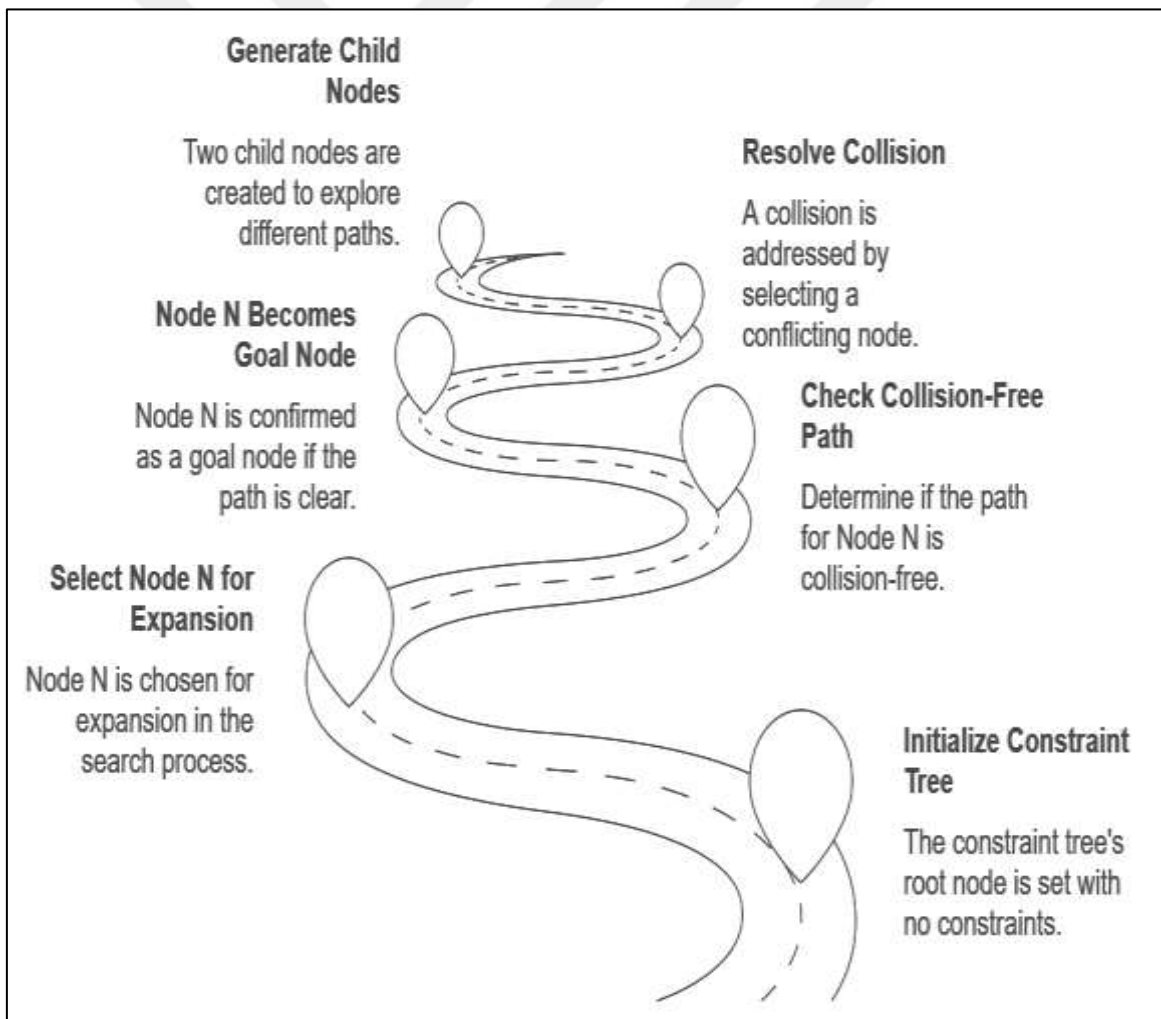


Figure 3.2: Conflict Resolution In MCBS.

The constraint tree's root node has an empty set of constraints, and a solution made up of the n shortest pathways. CBS determines if the solution is collision-free once node N has been selected for expansion. If it is, node N becomes a goal node, and CBS returns the answer. If not, CBS resolves the collision by selecting one of the conflicting nodes and updating Node N . For example, if CBS resolves a vertex collision between nodes a and b at time step t , only one of these nodes can be present in cell x at that moment. The restrictions are either a, x, t_i or b, x, t_i . CBS generates two child nodes. If CBS resolves an edge collision at time step t , it must consider restrictions such as b, x, y, t_i (preventing node b from migrating from cell y to cell x at time t) and a, x, y, t_i . During the low-level search phase, CBS swiftly determines the new shortest path for each node under the newly enforced constraints. For example, a vertex constraint prunes one node in the search tree, which may touch the pathways of other nodes but does not cross the environment's limits. Each child node must follow the limits provided. The distance between all robots and their individual objectives is then computed, with the closest position to each robot's goal selected. The distance ratio is useful in determining the optimal collision-free path. The proposed technique is used to address issues in warehouse packing, where several robots share the same environment but have different aims. To clarify further, suppose we have a constraint tree with each node representing a set of constraints needed to resolve conflicts during pathfinding. In this tree, the root node, known as the constraint's node N , starts with an empty set of constraints. Consider a scenario with two agents. The root node N is responsible for discovering the shortest pathways for both agents. Assume Agent 1's path is $[A, B, E]$, while Agent 2's path is $[B, C, D]$.

These pathways are estimated individually, with node N 's total cost equal to the sum of both agents' individual costs. Assume that the cost of traveling between nodes is uniform, with node N having a total cost of 4. at the simulation, notice that a collision will occur at node C at time step 1 when both agents try to occupy the same area. To address this issue, CBS will divide the root node N into two child nodes, each with a unique constraint to prevent collisions. A restriction is imposed to the left child node to prohibit Agent 1 from existing at node C during time step 1. The restriction is expressed as $[1, C, 1]$. After introducing this limitation, CBS recalculates Agent 1's shortest route. The revised path may be $[A, A, C, E]$, with the agent stopping at node A for a one-time step before continuing. This new path now costs 3 (including wait time). Agent 2's route remains unaltered since it does not need to avoid node C . The total cost of this child node is 5 (path cost plus wait time). A new

restriction is applied to the right child node to prohibit Agent 2 from existing at node C at time step 1, denoted as [2, C, 1]. Again, CBS recalculates the shortest path for Agent 2, which may be [B, B, C, D], with a wait time at node B. Agent 2's fee for taking this new path increases to three. Agent 1's route is the same as in the root node. This child node has a total cost of 5. In this example, the CBS algorithm generates two child nodes with the same total cost of five. However, the algorithm will choose the node representing a collision-free path as the target node. In this scenario, because both pathways have distinct requirements that fix the collision, one might be chosen based on other variables such as search algorithms or extra constraints. The base of the CBS method is the systematic way in which it enhances efficiency in pathfinding within multi-agent settings. In implementing CBS, specific collision-prevention constraints are incrementally added for each agent, ensuring that such sets of requirements are increasingly incorporated into the pathway calculation. This carries out a strategic exploration of possible routes through continuous recalculation of the paths while considering these limitations. Finally, it selects that path which gives a minimum travel cost and successfully avoids collision between moving agents. The complete process of constraint-based decision in CBS is represented diagrammatically in Figure 3.3.

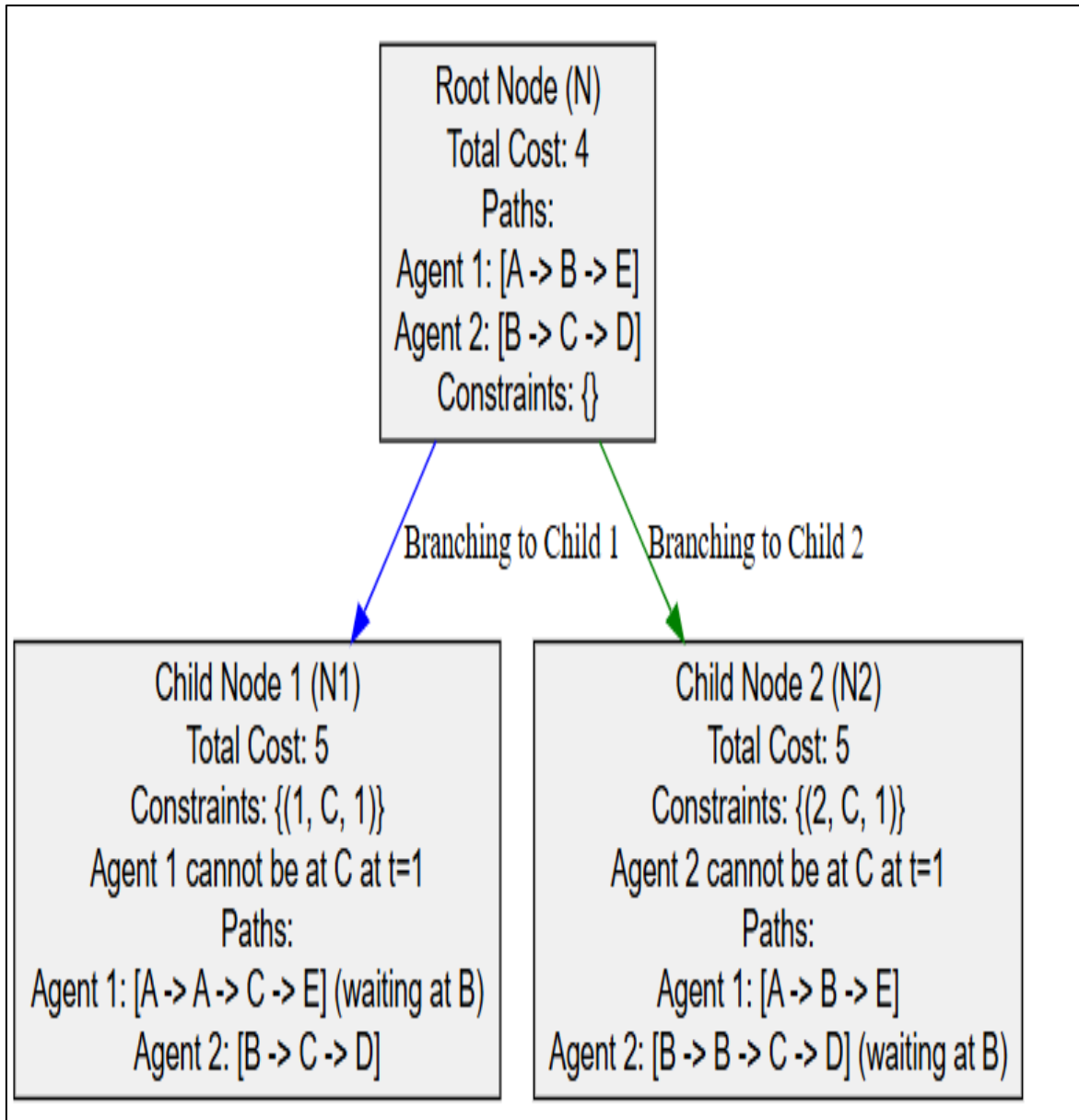


Figure 3.3: Example of Tree Node of The Multi-Agent Path.

The key to CBS is that it picks at any one time, the least expensive node to expand-the node which most advances agents toward their goals. If there are two nodes which would cause a collision, CBS splits the parent node into two separate child nodes, as previously mentioned; each of those child nodes includes additional constraints designed to resolve the identified conflict. This is known as node expansion and is done only when there is the prediction of collisions. In this method, the algorithm enters into the depth search for all nodes generated through a collision scenario. It assumes that in every time step, each agent is associated with one vertex; hence, the algorithm efficiently handles multi-agent's entanglement with the

environment. This assumption is practical and drives the algorithm to limit the number of constraints it has to apply; approximately $O(nkC)$, with n , being the number of agents, k is the number of time steps, and C is the number of constraints. Because CBS controls this number of constraints wiser, the algorithm can handle and resolve collisions much more cleverly in a way that quickly leads toward finding a feasible solution for all the agents. Each node in the constraint tree is often constrained by every constraint, and so, in general, the total amount of constraints constrains the depth of the tree.

The CBS time complexity is hence overall bounded by: $O(nC \cdot 2^{knC})$, where n denotes the number of agents, C represents the number of constraints, k is the number of time steps, and 2^{knC} describes the size of the constraint tree. accounts for exponential growth of the possible combinations of constraints. As already pointed out, prioritization is vital in conflict resolution and prevention of collision. Suppose an agent is stuck because of priority problems. Let a low-priority agent hold a resource that another high-priority agent needs. This will surely lead to suboptimal solutions or even failure, considering the case of a medium-priority agent. This again points to the importance of good priority management, whereby higher-priority agents could access resources in case of need, without unnecessary delays and conflicts. In case of a priority conflict, an agent that has inherited its priority from another one of higher priority must wait to see if the situation will resolve itself successfully.

In case it does work by waiting and resolves the conflict, then the agent may continue toward its destination node. If it doesn't work, however, the agent must detour onto an alternative path, keeping clear of nodes already occupied or to be occupied by agents of higher priority. This is a priority-based approach that enables higher-priority agents to have a high possibility of successfully passing through to reach their goals, while the lower-priority agents route their paths in such a way as to avoid collisions and resource conflicts. Convergence continuously takes place through these priority constraints by the algorithm to generate collision-free paths that efficiently guide all agents to their respective destinations. The Conflict Tree at the beginning does not carry any constraints at its root node. But, as the algorithm expands, it proceeds towards adding constraints so that the conflicts may be resolved, and every agent navigates smoothly. This lack of constraints enables the free computing of alternative pathways for all agents. The root node serves as the basis for developing the CT, directing the search for a collision-free path. In conclusion, the CT's

function in CBS is to carefully investigate and assess viable pathways while resolving any conflicts that develop. The CT's root node, which begins without restrictions, gradually expands as conflicts are identified and resolved.

When no further conflicts are found, the CT node is selected as the destination, indicating a successful solution. R_i is a mobile robot where $i=1$, to n , that represents the agent ID this representation is done using Eq. (1).

$$C_i(t) = (x_i, y_i, t) \quad (6)$$

If there is a collision between R_i and R_j , the result will be as Eq. (2).

$$\text{Collision } (i, j) \rightarrow \exists t: C_i(t) = C_j(t) \text{ where } i \neq j \quad (3.8)$$

The maximum path will be selected after producing the length of the path (μpl) as in Eq. (3).

$$\mu pl = \sum_{i=1, j=1}^n p^{ij} \quad (3.9)$$

The following equation can be used to express these scenarios numerically to ensure that the solution is optimum. Presenting the free routes after getting the total number of agents inside the map, and then reaching the goal, Eq. (4).

$$\text{Collision } (i, k) = \phi \forall R_i, R_k \text{ where, } i \neq k \quad (3.10)$$

First deleting the conflict mobile path if that happened. Second, pl ought to be at a minimum. Consider the following while employing the specified set of ideal pathways (S) for all agents inside the device:

$$S = [P_{ij, \dots}] \forall i, j \text{ where } D_{i,j} = 1 \quad (3.11)$$

and $\min(\mu pl)$ and $\text{collision } (i, k) = \mu$ for each pair of R_i, R_k robots in the system.

3.4 GENERAL PATH METHOD

First of all, the simulation process will transform the warehouse environment into some graph based on nodes. This step is a prerequisite for the application of the pathfinding algorithms that allow orderly analysis and routing inside the warehouse. Each node in such a graph depicts a certain location or area in the warehouse, while edges represent possible

movements across those locations. Then, the source-to-destination pairs are matched by taking the ratio of the distance of each destination assignment. This method still allows the determination and selection of the most efficient paths in which the robots will travel, considering both distances and spatial limitations of the warehouse. The work develops collision-free paths using a distance metric-based approach.

This approach ensures smooth operations without collisions. It will calculate the best routes by knowing the spatial relations among robots and obstacles, hence ensuring that each robot will be able to pass through the warehouse without interference with other robots. The avoidance of collision is further enhanced by the use of the reserved tables that allocate different paths for each robot. These tables govern and avoid mutual collision, guiding on the thought that two robots cannot be in one place at one particular time. The system minimizes the chances of a conflict by dedicating the path so that the robot can continue further along its path without dispute. The environment is modeled with a habitat comprising a number of robots and an input reference map. A map defines accessible areas and obstacles inside the warehouse. The reference map is essential for modeling very well the operating limits and space constraints that need to be obeyed by robots. The system can then test and optimize robot behavior in a controlled yet realistic environment by simulating various scenarios and changing pathfinding and collision avoidance strategies. This will improve the performance of actual warehouse operations.

The robots begin from any specific given position within the warehouse. Then, there is a need to assign goals to each one of them through an intelligent selection technique. It would provide the goal-mapping matrix, indicating the best goal for every robot concerning variables like distances, priorities of tasks, and operational constraints, thus guiding the path-planning process. For better pathfinding efficiency, the ratios of distances for each target are calculated. It helps in giving an estimate of how much open space each robot will travel through and enables the algorithm to choose a path such that travel distance is minimized while bypassing obstacles. With this analysis of distance ratios, the system would distribute the paths more efficiently for effective navigation. In a static warehouse setting, each and every transfer or movement has a cost assigned to it with an intent to make sure the paths chosen will be cost-effective and efficient, considering the distance of travel, amount of energy consumed, and time expended by each movement.

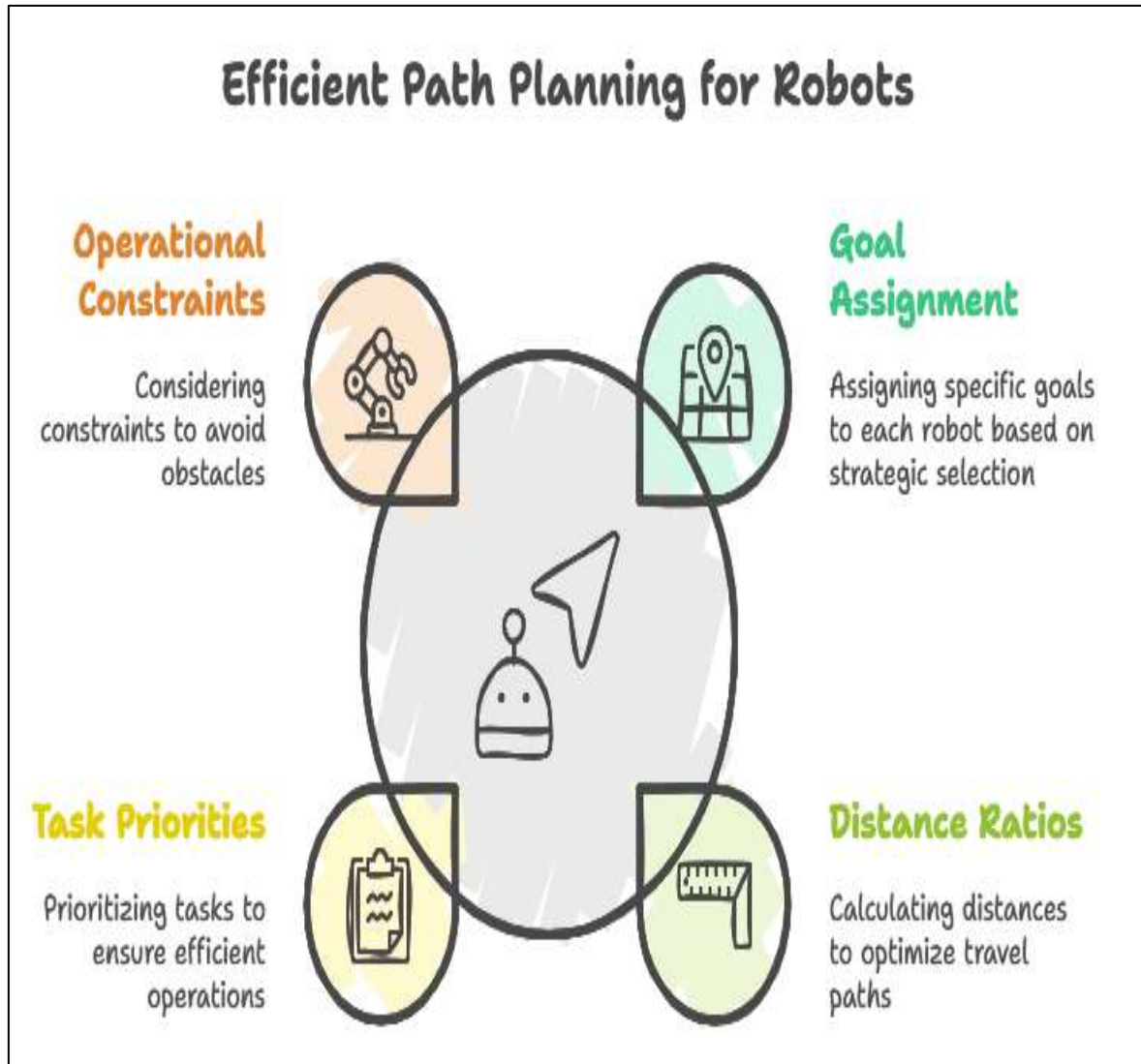


Figure 3.4: Efficient Path Planning for Robots.

With this analysis of cost, the system can pick the best route for each robot and carry out its task as quickly as possible while attempting to reduce operational expenses. The flowchart shown in Figure 3.5 describes the modification of the work structure.

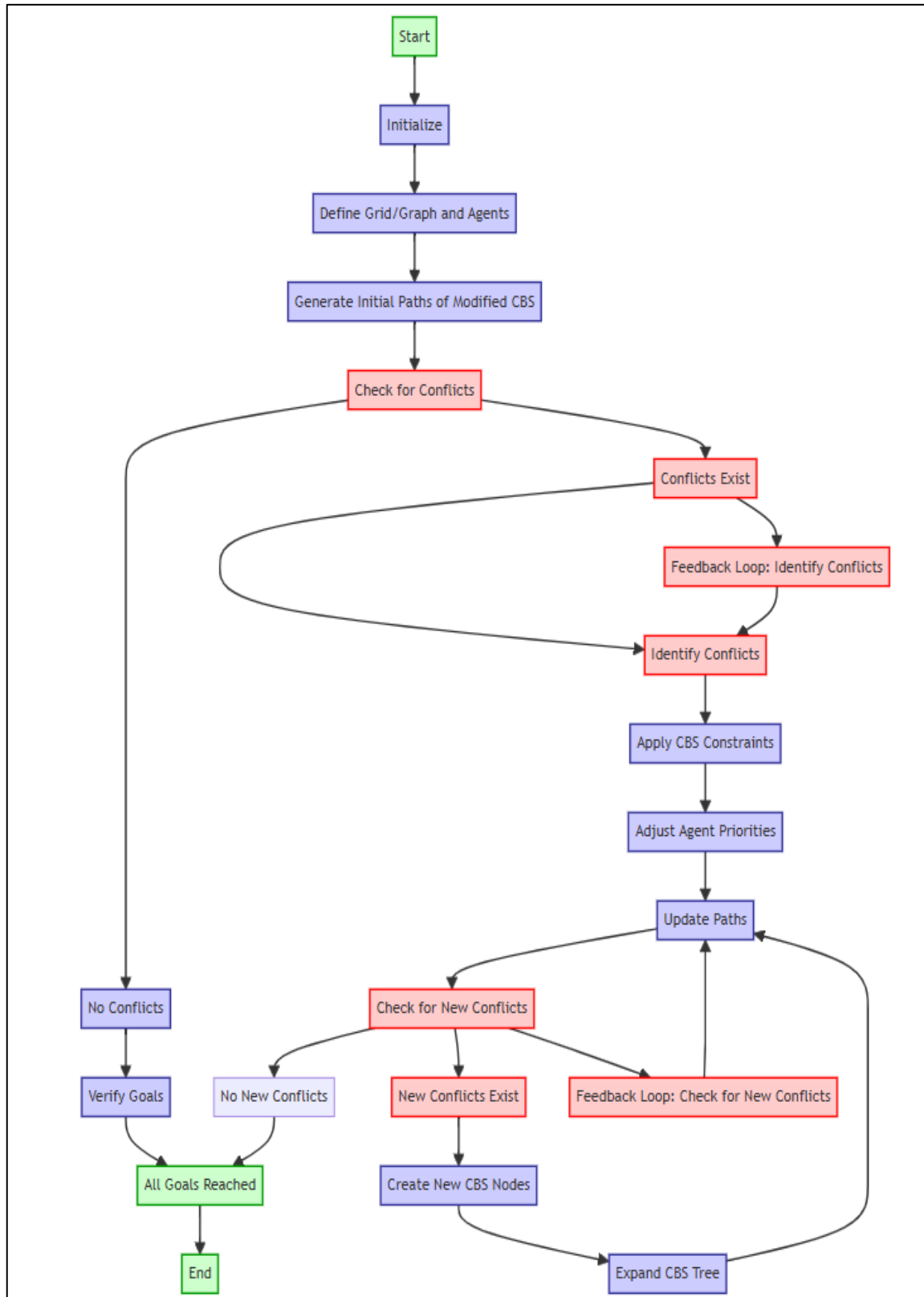


Figure 3.5: Shows The Work Structure.

4. SIMULATION STRUCTURE

4.1 IMPLIMENTATION SETTINGS

The efficacy of the developed technique in planning the routes of multiple robots is evaluated through several scenarios that include fixed and dynamic obstacles with known and unknown environments. In dynamic environments, each robot has to head towards their target while considering the other robots as moving obstacles. In such scenarios:

- Red Circles represent the robots.
- Green Circles represent moving obstacles.
- Black Blocks show permanent barriers.

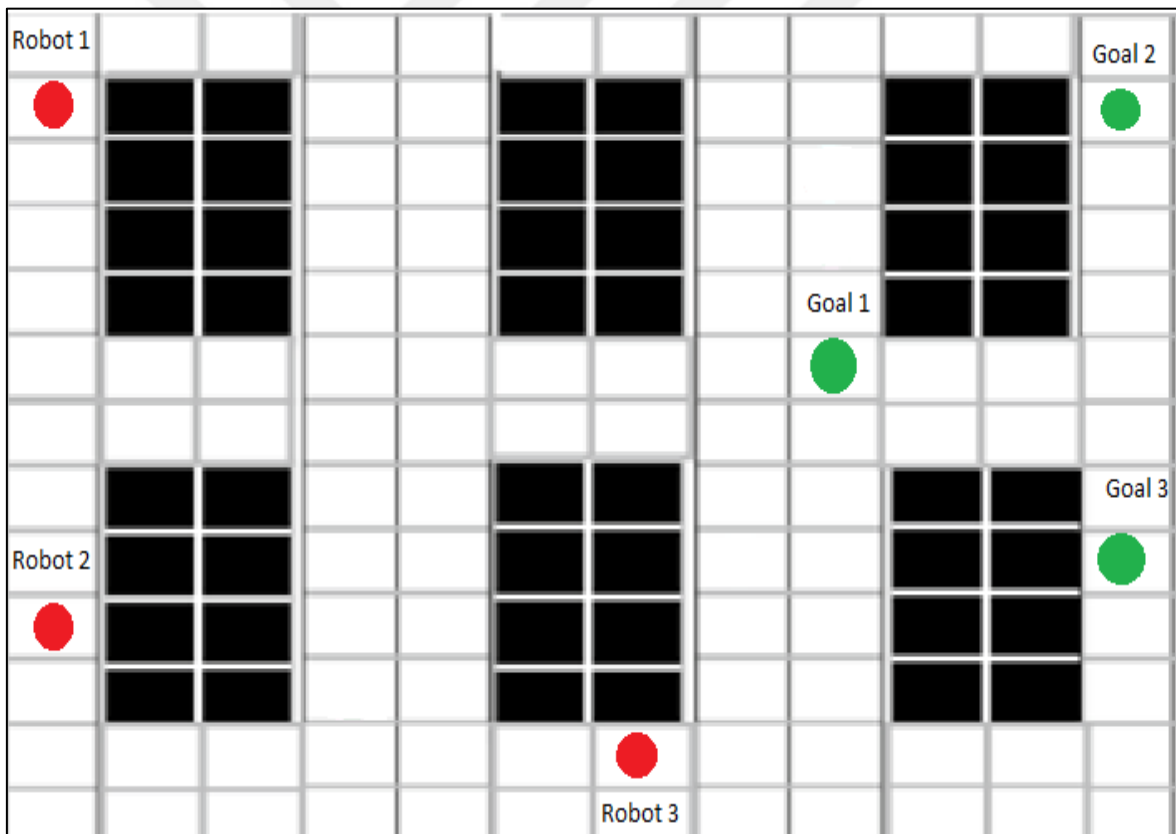


Figure 4.1: Environment Setting.

Target assignments for robots are made with evaluations of the cost of every possible path that can be taken in both known and unknown environments. Assessment of cost in this context involves:

- a. Total Path Length: Distance over which the robot will need to travel.
- b. Number of Collisions: Frequency of collisions by the robot against obstacles or other robots.

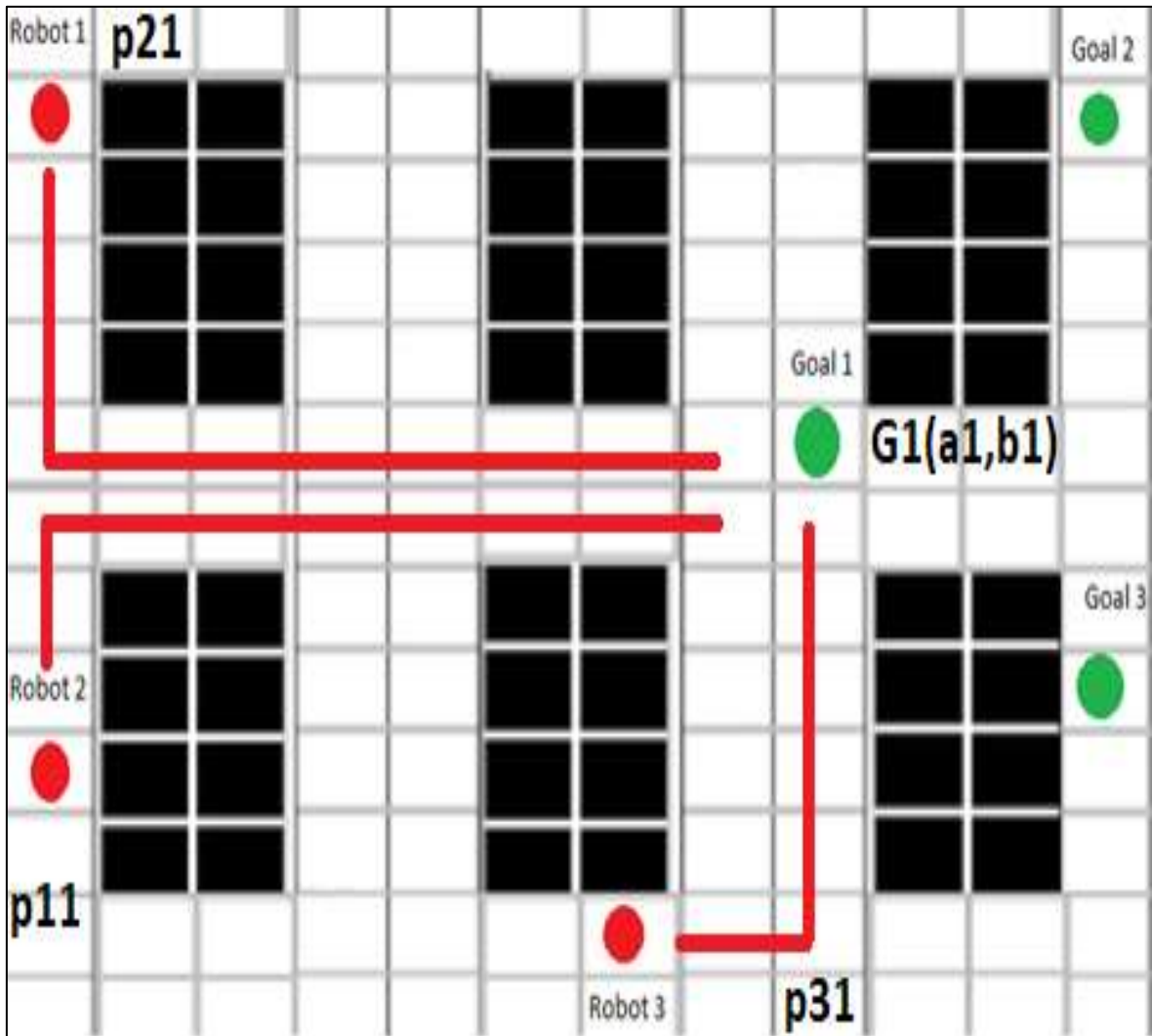


Figure 4.2: Each Agent Setting Goals.

A cost function such as this can be helpful in the target's assignment of a group of mobile robots by minimizing these costs for efficient and collision-free routes. Target destinations are usually at random in unknown or new environments; this complicates the process of path planning. This approach will provide the system with the ability to adapt to different conditions and cope with navigation from robots in both situations. To be complete and optimal, we need to prove that every iteration of the top layer of our method brings us closer to the solution.

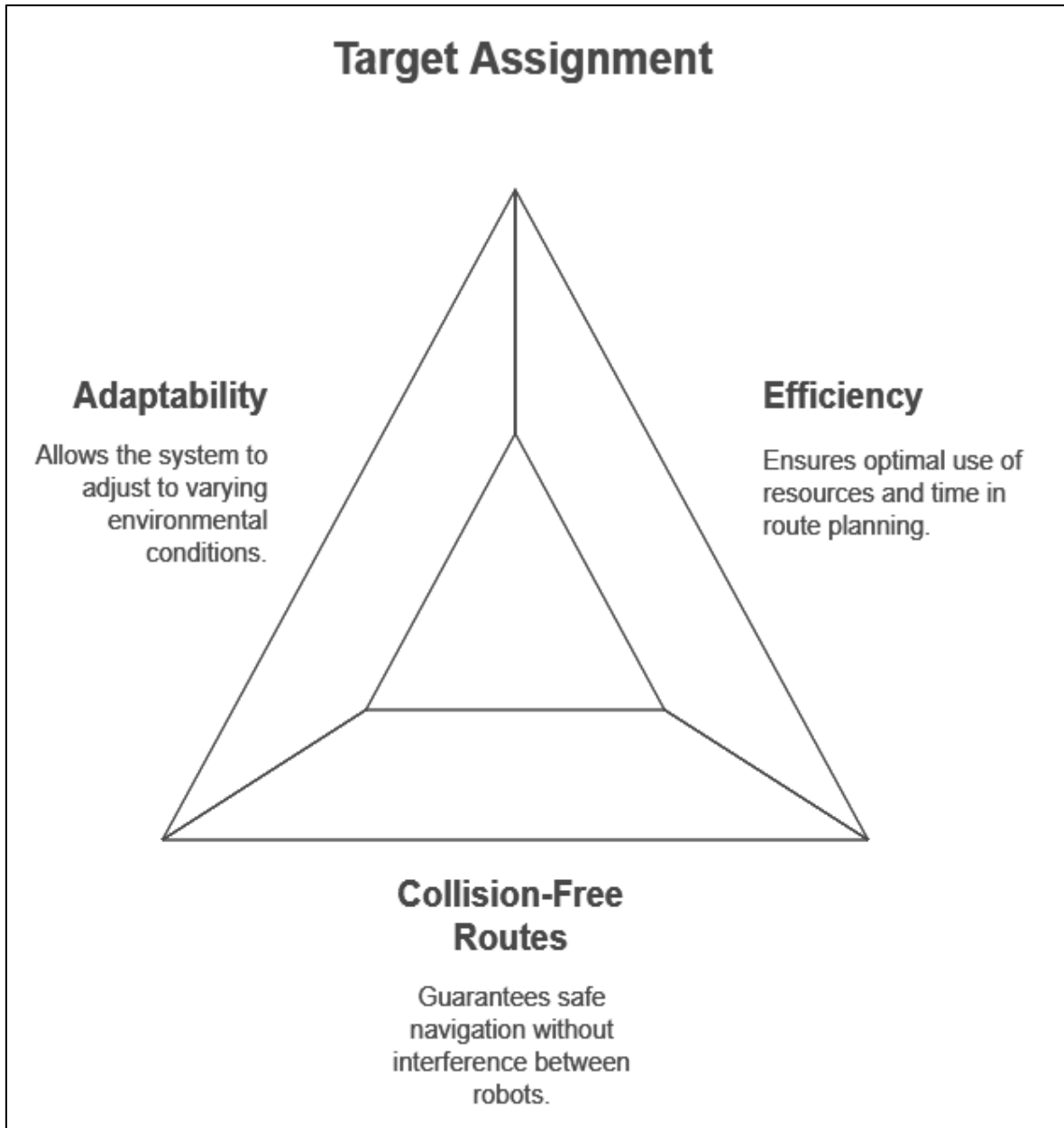


Figure 4.3: Target Assignment Map.

The algorithm first starts with a full multiverse that holds all the possible multiverses and hence all possible solutions. As the algorithm develops, we will generate a sequence of multiverse sets called the working sets. Each set in this sequence will further shrink the search, removing multiverses that produce collisions with the current working set. Each of the working sets within this series is related by the principle of a branching set, where one multiverse is simply changed out for a different, more refined set. The iterative refinement of each set means that it deals with the residual or remaining conflicts and constraints of the

original multiverse. As it goes through the chain of narrowing possibilities and discarding infeasible solutions, the algorithm progresses towards an increasingly precise and optimal solution. In short, this process is iterative and ensures that the entire gamut of solutions is considered and iteratively refined, with the completeness and optimality of the approach thus demonstrated. Let's consider a target assignment example that shown in Figure 4.4:

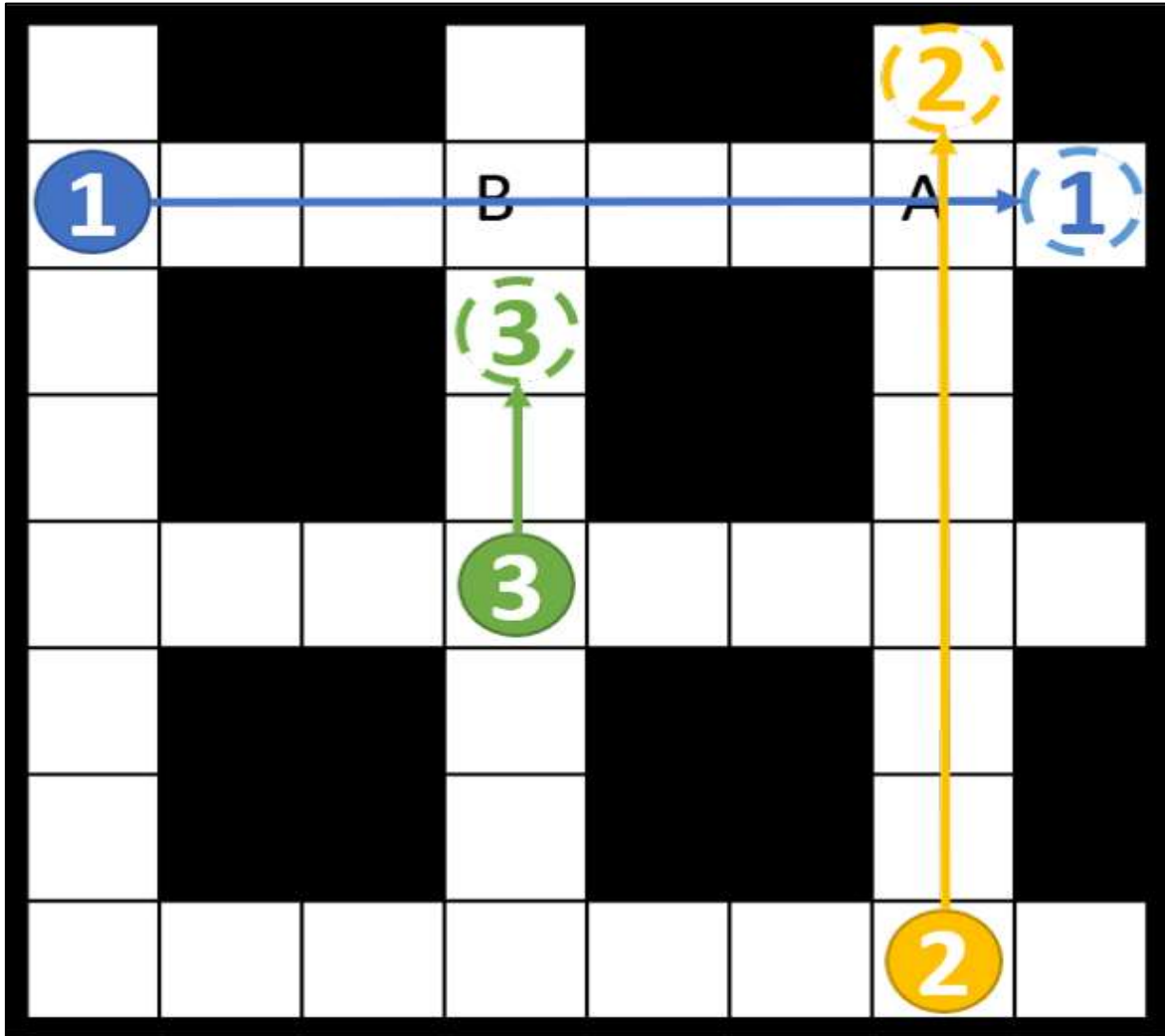


Figure 4.4: Shows Example of Target Assignment Within the Environment [45].

4.2 EXPERIMENTS SCENARIOS

Quantifying the performance of the proposed strategy, this study constructed approximately up to 230 different maps, each designed to simulate a wide range of motion scenarios and obstacle configurations. They ranged in size and shape, with obstacles forming various concave, rectangular, and circular forms. The simulation was divided into three discrete

scenarios, each at a different level of difficulty. Testing on these varied maps and scenarios evaluated the strategy through a total of various challenges and environments so that no aspect of strategy performance would be left out in its capability to navigate and solve pathfinding under different situations.

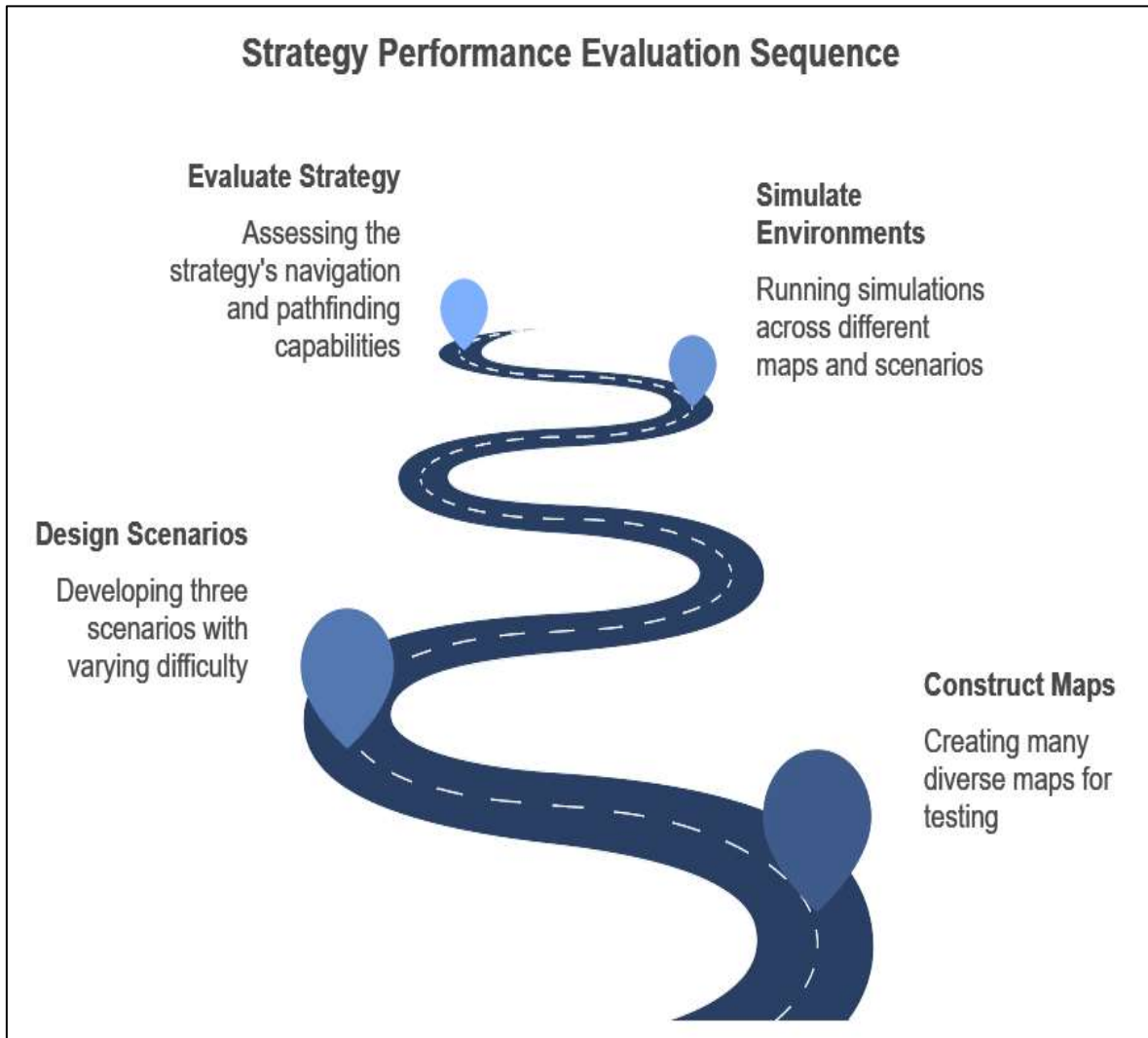


Figure 4.5: Experiment Scenario Process Map.

4.2.1 First Scenario

The first test case was of medium difficulty. It included a mixture of terrain and positions and shapes of obstacles. For this case, agents were put into previously unseen environments and were facing multiple dynamic obstacles. In such cases, the agents were supposed to navigate through these obstacles to their goals without touching them. This scenario is designed to test the capability of the strategy in handling moderately complex environments,

where the layout and obstacles would change, hence creating near real-world conditions that require effective pathfinding and collision avoidance.

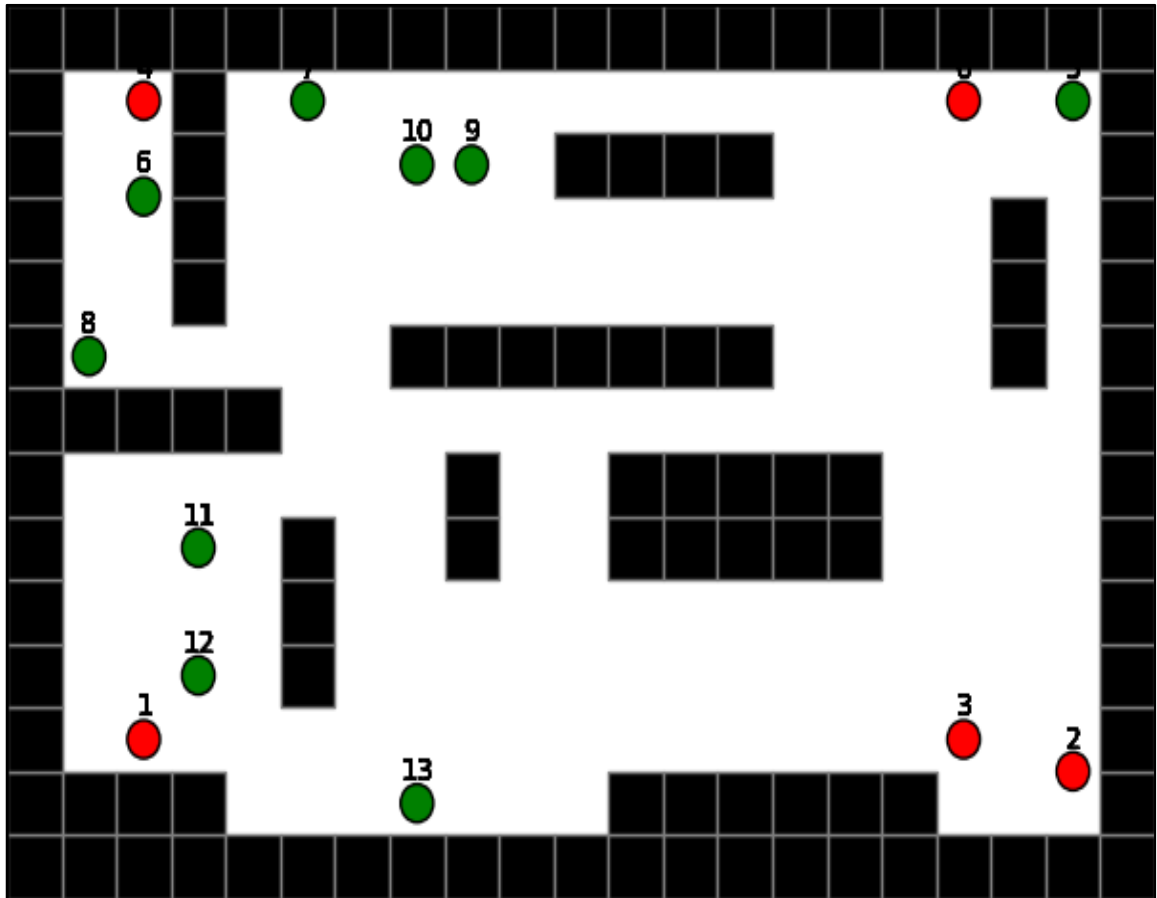


Figure 4.6: First Scenario Example.

Figure 4.6 illustrates one result of path planning run on an environment with dynamic targets for which no prior knowledge about the specifics is assumed. The objective is that in this setting the robot will finalize its tasks while successfully avoiding collision with fixed and moving obstacles. Under such a scenario, the proposed technique provides the best path planning for the robot, closest to its target, where other crucial factors such as travel time and path length have been taken into consideration for easy and safe navigation of the robot. In this regard, the current approach focuses on ensuring minimum travel time while considering the overall distance covered for efficiency in the operation within dynamic environments. Knowing the initial and goal positions of each robot, the algorithm has to calculate the journey length that each path has been required to do in order to check whether such a path is effective or not. The calculated journey length helps with the determination of

feasibility and suitability of the path in respect to the requirements of the robot's navigation. Table 4.1 summarizes the results of experiments. Included in this table are:

- a. Starting and Goal Positions: Initial and target locations within the environment for each robot.
- b. Time Expected to Achieve Goals: An approximation of how much time each robot needed to achieve its goals.
- c. Path Lengths: The length of the path each robot travelled through to reach its destination.
- d. Agents Involved: The number of robots participating in this pathfinding.
- e. Size of Maps: The size and complexity of maps on which this algorithm was tested.

All this information together allows the in-depth analysis and comparison of any path-planning strategy's performance under a lot of different conditions. This emphasizes how well the technique copes with different factors, obstacles, and dynamic changes in order to realize successful navigation performance in complicated environments. It pinpoints that, in order to achieve successful navigation performance, collision avoidance and the use of priority rules are an absolute must in cases when multiple robots act together and share the same space. This is a priority-based approach that allows robots to coordinate their movements efficiently and minimize conflicts in an optimum path through the environment.

Table 4.1: Result Description of The Proposed Approach.

Start	Goal	T	PL	M	Map
(1, 1)	(12,18)	5.4	4.9	8	8 * 8
(11,9)	(6,1)	2.3	3.3	8	8 * 8
(12, 6)	(6,10)	5.3	3.5	8	8 * 8
(11,2)	(2, 18)	6.1	4.3	8	8 * 8
(1, 17)	(11, 1)	4.7	3.5	8	8 * 8
(12,12)	(1,2)	3.4	6.5	8	8 * 8
(12, 13)	(2, 6)	6.6	7.2	8	8 * 8
(11, 19)	(3, 10)	3.9	4.7	8	8 * 8
(4,1)	(6, 19)	2.6	5.2	8	8 * 8
(7, 19)	(4, 1)	3.2	4.5	8	8 * 8
(9, 19)	(4, 2)	4.3	2.4	8	8 * 8
(1,10)	(12, 16)	5.5	4.2	8	8 * 8
(1,15)	(12, 5)	6.9	5.3	8	8 * 8
(1,5)	(10, 12)	3.2	2.4	8	8 * 8
(1,4)	(12,1)	3.0	4.2	8	8 * 8
(6, 1)	(1,5)	3.4	2.7	8	8 * 8

The success rate for destination arrival was documented at 6.9, showing the effectiveness of the mobility of the robot in an unfamiliar area. A number of factors would produce deviations in the success rate of the robots, including time taken to avoid collisions and any delays caused by priority rules when robots are crossing each other's path. Effective collision avoidance and priority determination are a big basis for performance analysis in the path-planning strategy. An example is when there are two robots moving in the same direction and reaching a spot where there is just one exit. The priority mechanism makes sure the robot that is closer to its destination is given the chance to go through first. It is only after the first robot has been able to cross through the exit that the second takes a follow-through action. The scenario result will be shown in figure.4.7. The following diagram describes the movement of agents within the simple environment from their starting points in order to reach the goals. It also shows us the moving obstacles that are shown in the environment.

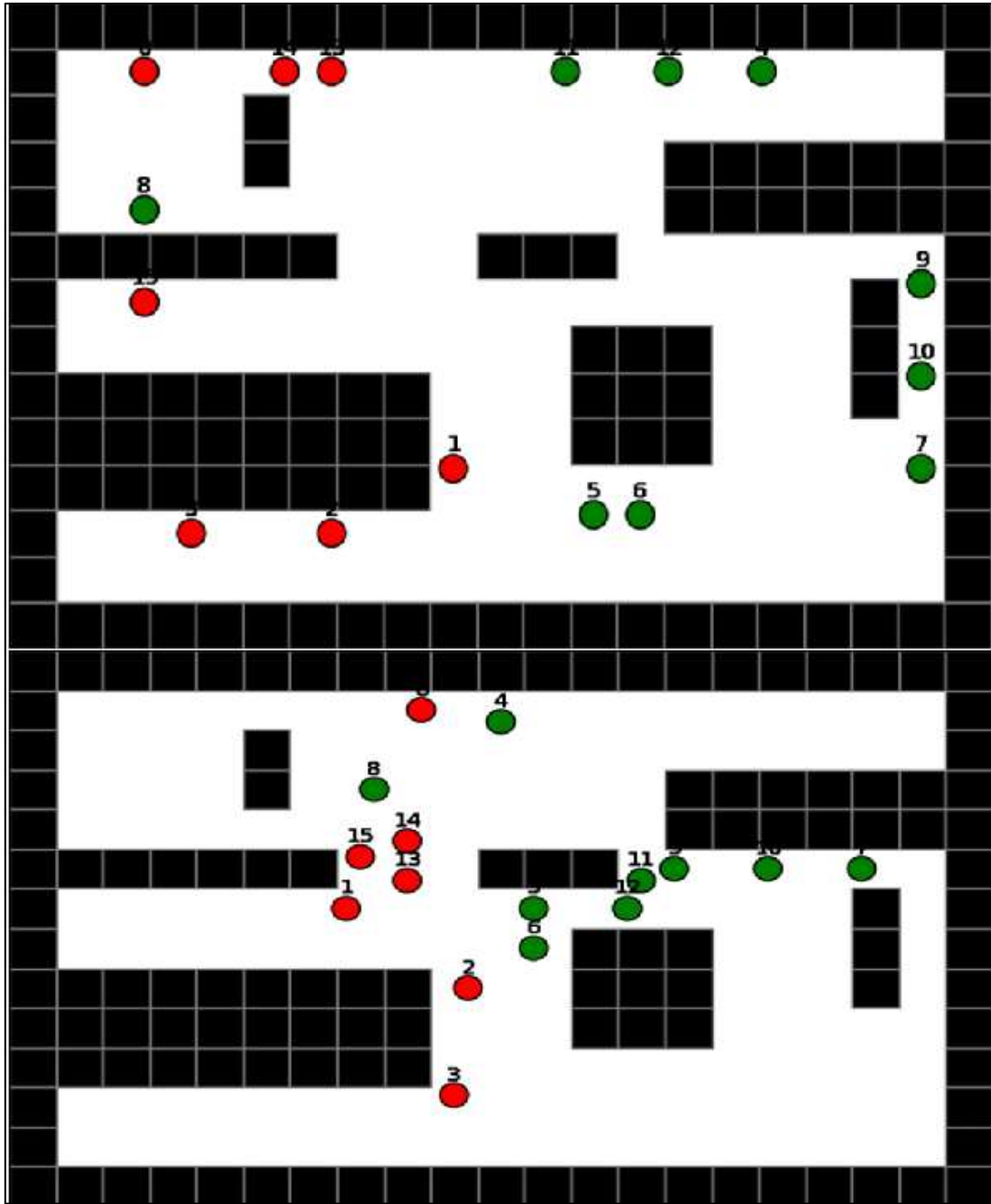


Figure 4.7: Example of the First Scenario as A Simple Environment.

Through figure 4.7, we can see the beginning of each agent's departure from his original location in order to reach the target without colliding with any obstacle in the environment. The next result will show another example of the first scenario.

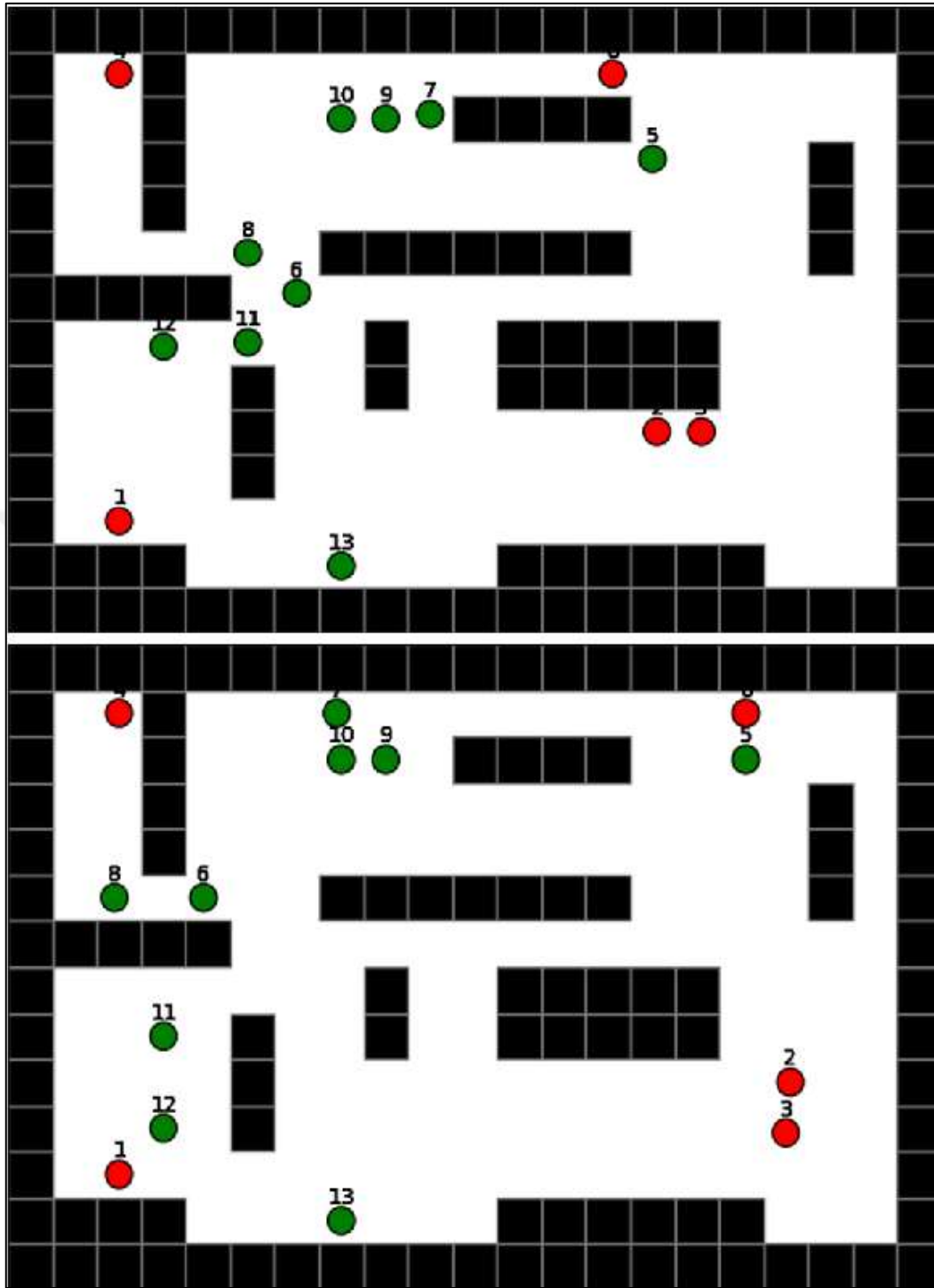


Figure 4.8: Another Example of The First Scenario.

As shown in figure 4.8, The results also show that the clients try to reach their targets without colliding with fixed or moving obstacles in the environment.

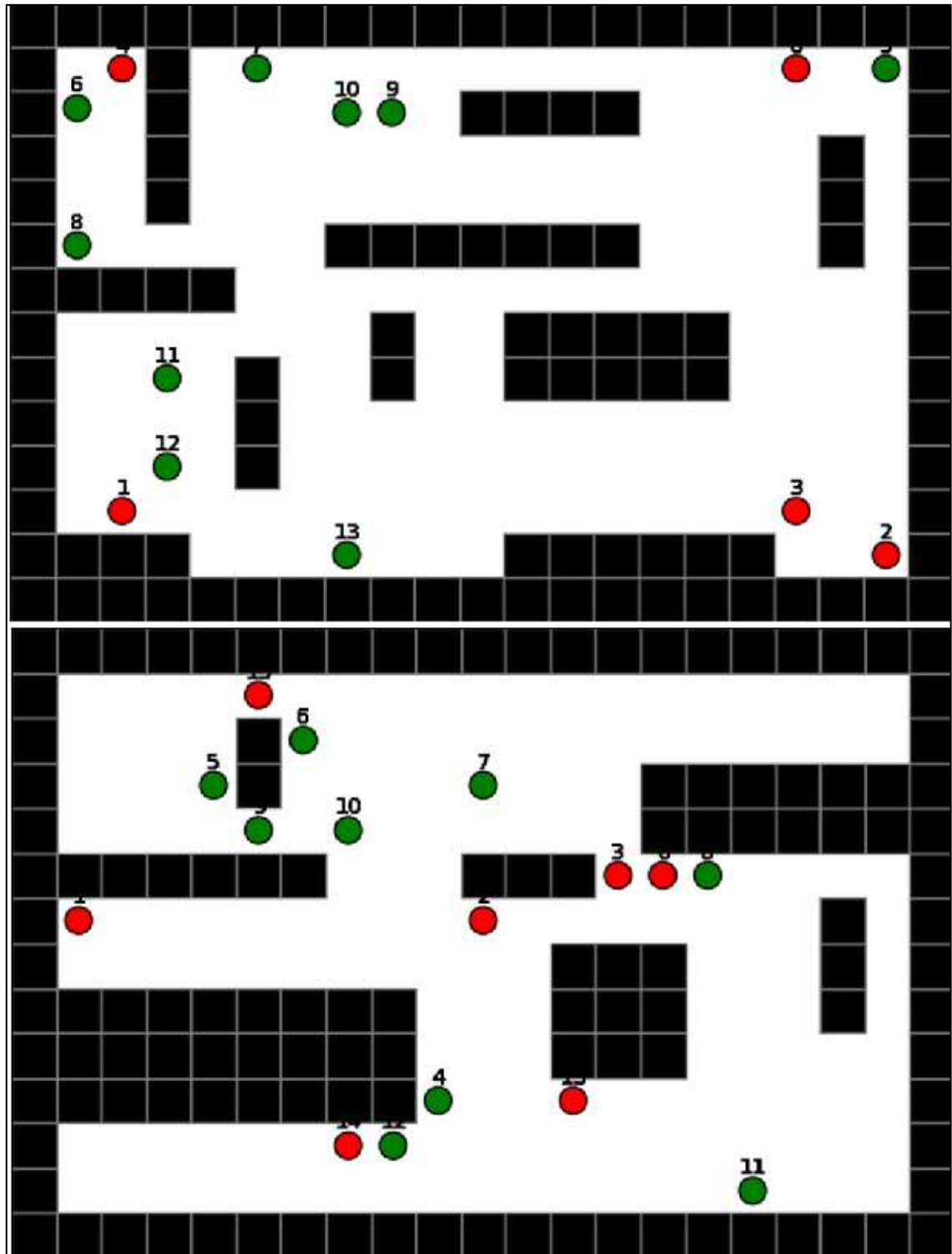


Figure 4.9: Simple Example About the First Scenario.

Through the results obtained by figure 4.9, we find a summary of the clients' attempts to reach the goals from the moment they start until they reach the goal.

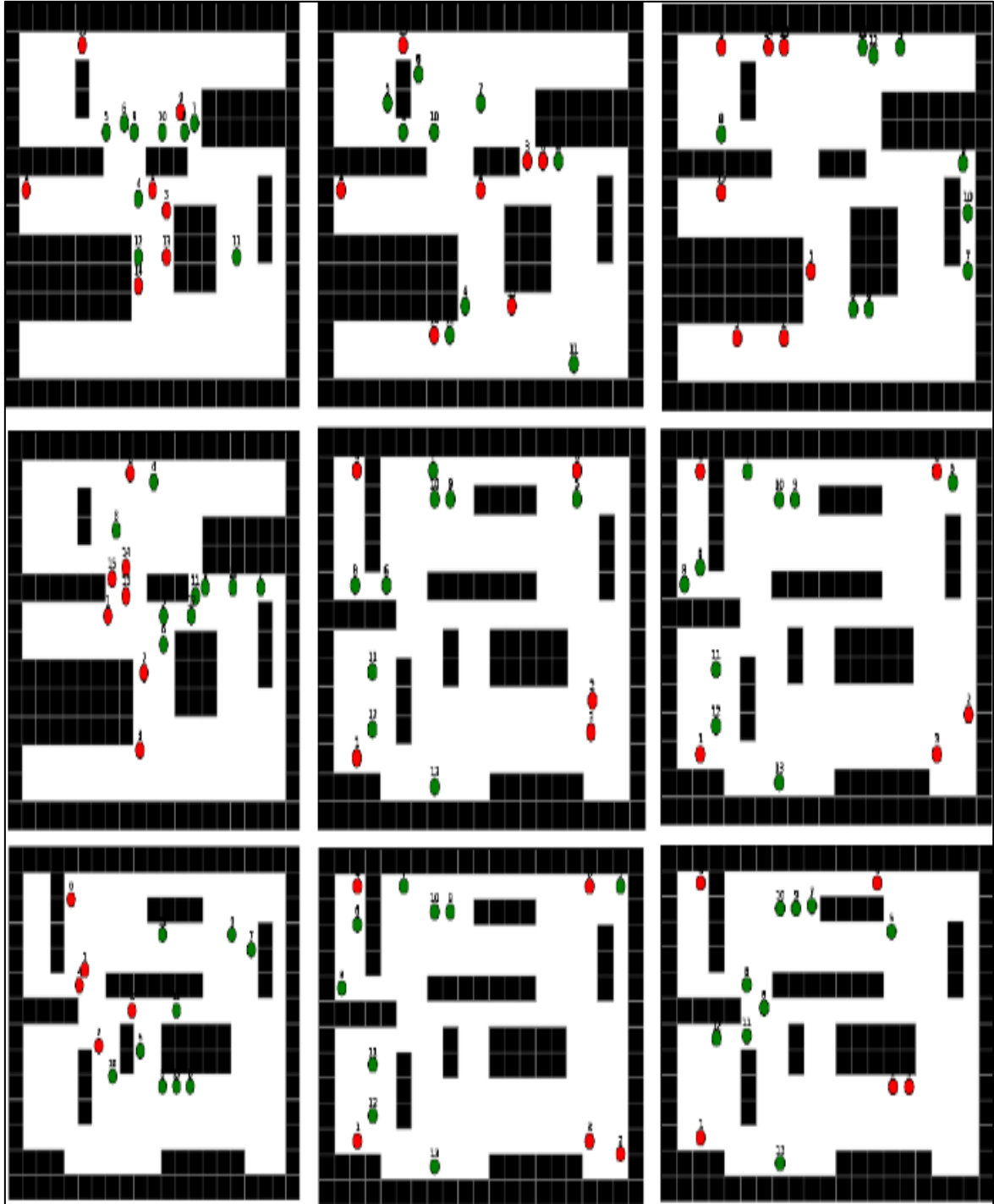


Figure 4.10: Illustrate A General Example of Different Experiments.

The results obtained in figure 4.10 show us the agents trying to reach their goals without colliding with obstacles. The results show a set of tests that were conducted on the first environment, which is considered simple, in order to try it out and ensure that it is free of any errors.

4.2.2 Second Scenario

The improved multi-agent pathfinding methodology is subjected to a well-regimented test on a range of map metrics made up of static and dynamic obstacles. Other robots act like moving obstacles, with one robot acting as the main agent. The idea is to ensure that all agents safely reach their destinations without collision with static obstacles or other agents. For proper testing of the effectiveness of the algorithm, the map is expanded progressively by adding new dimensions and enhancing its complexity. This entails making the size of the map larger in dimensions to accommodate more complex environments and additional robots in order for its scalability to be tested when there are more agents involved, plus increasing obstacles that would give the algorithm a hard time avoiding collisions. It also evaluates the algorithm in unseen environments with dynamically changing obstacles, possibly modelling real-world situations where some form of randomness is to be expected. This method ensures that the performance of the algorithm will be tested thoroughly and that it will aptly demonstrate capability for the algorithm to manage pathfinding under different conditions and progressive complexities, Figure 4.11 illustrates the new environment setting.



Figure 4.11: Second Scenario Setting.

Figure 4.11 emphasizes the changes in dynamics at the workplace and how that can impact the navigation of robots. This figure depicts how factors in the surrounding area drive environmental changes and how the robot adapts to those changes. In the wake of increased complexity within the environment, previously feasible paths to the destination may disappear or become obstructed; in turn, this shows how continuous path readjustments may be necessitated or even unavoidable. Changes like adding obstacles or changing the paths update the planning calculations done by the algorithm. The environmental changes - such as adding static obstacles - significantly alter key performance metrics like path length, travel time, and the success rate in reaching the destination. Table 4.2 provides the breakdown of these performance indicators for the various scenarios in updated path lengths, travel times, and success rates. This table will help gain insight into how well the algorithm adjusts to evolving conditions and maintains efficient and effective pathfinding without losing much performance.

Table 4.2: Simulation Result Under Dynamic Movement.

Start	Goal	T	PL	M	Map
(1, 7)	(5,23)	7.5	7.9	16	32*32
(11,7)	(3,27)	3.9	5.4	16	32*32
(5, 3)	(15,18)	9.3	12.5	16	32*32
(8,14)	(21, 25)	10.1	6.3	16	32*32
(8, 7)	(31, 24)	7.7	5.5	16	32*32
(1,5)	(29,25)	6.4	6.3	16	32*32
(6, 10)	(32, 28)	8.6	4.2	16	32*32
(15, 7)	(30, 32)	5.9	6.7	16	32*32
(18,7)	(26, 29)	2.6	4.2	16	32*32
(32, 10)	(28, 30)	4.2	4.5	16	32*32
(13, 12)	(19, 18)	3.3	5.4	16	32*32
(11,1)	(30, 4)	2.5	4.2	16	32*32
(24,10)	(32, 6)	7.9	5.3	16	32*32
(20,12)	(31, 10)	5.2	3.1	16	32*32
(7,1)	(23,10)	4.0	2.2	16	32*32
(13, 18)	(32,12)	3.4	4.3	16	32*32
(18,8)	(25,10)	9.5	5.2	16	32*32
(23,6)	(20,23)	4.5	4.5	16	32*32
(23,15)	(19,12)	3.2	3.4	16	32*32
(22,17)	(18,11)	2.2	6.1	16	32*32
(29,11)	(15,19)	7.6	7.1	16	32*32
(23,10)	(15,21)	6.9	6.2	16	32*32
(27,8)	(6,10)	10.5	10.6	16	32*32
(26,15)	(1,17)	9.3	9.2	16	32*32
(27,12)	(1,11)	8.1	12.1	16	32*32
(30,25)	(10,30)	7.6	8.2	16	32*32
(31,28)	(10,11)	6.8	7.4	16	32*32
(30,2)	(2,17)	4.6	6.6	16	32*32
(26,15)	(1,20)	3.5	3.4	16	32*32

Based on the results obtained and also in Table 4.2, a number of inferences regarding pathfinding algorithms may be ascertained. First, an increase in the environment's complexity brings about increased time to the goal taken, coupled with a rise in the length of the journey of the robot. While it may not be a dramatic increase in time, the increase points to the complexities of route navigation in dynamically changing barriers but, more so, in a setting where many factors affect an environment. These results point out how pathfinding is time-consuming under complex and variable environments, hence justifying the need for adaptive strategies in shifting obstacles toward maintaining optimal navigation. Figure 34 will describe the overall changing as follow:



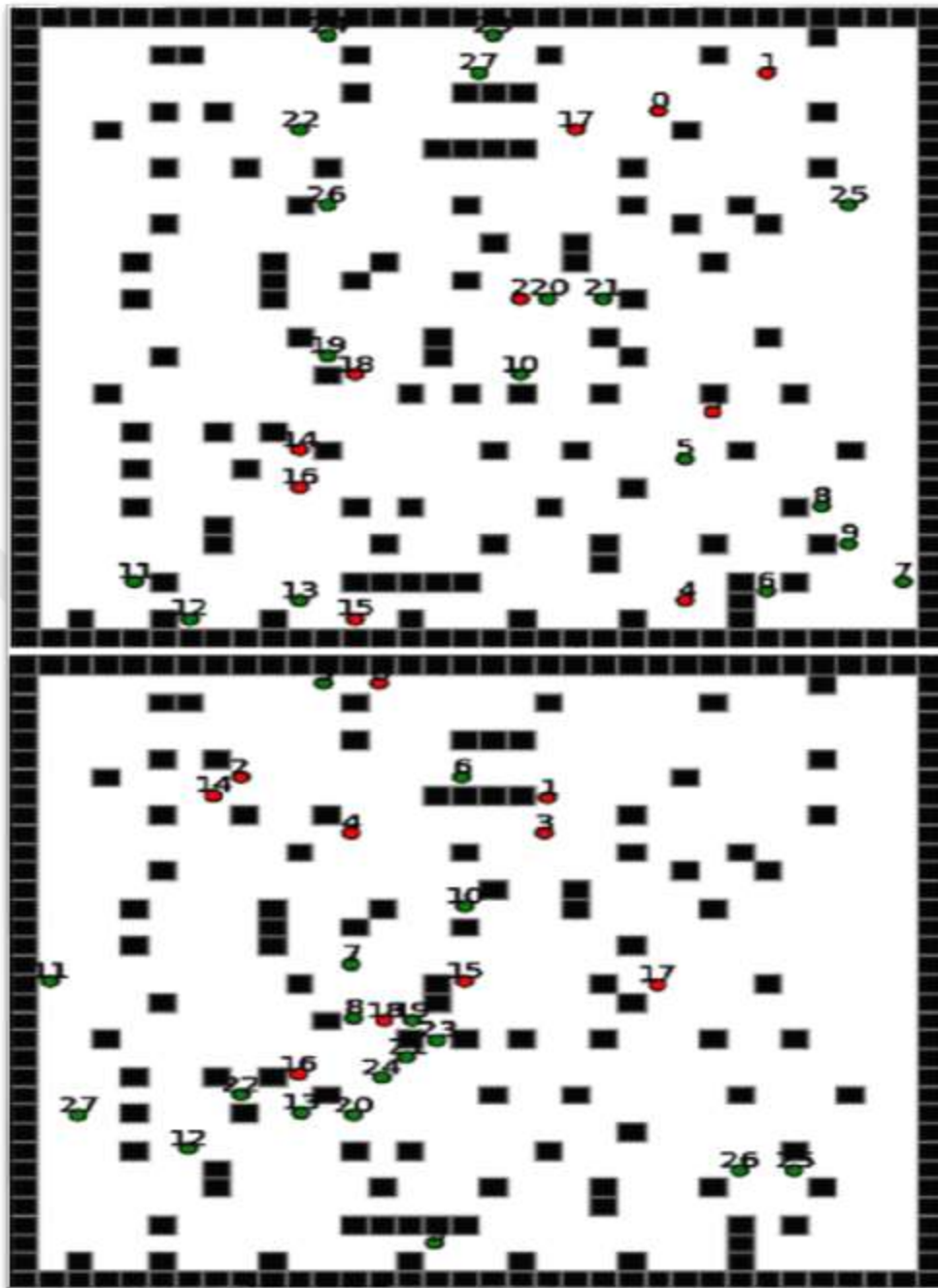


Figure 4.12: Illustrate The Changing in The Environment Complexity.

The result shown in Figure 4.12 shows us the increasing complexity of the environment by adding a number of obstacles of various shapes that are considered fixed, as well as obstacles that move in succession with the clients in order for them to reach their goals. Next figure will show another example of the complex environment.

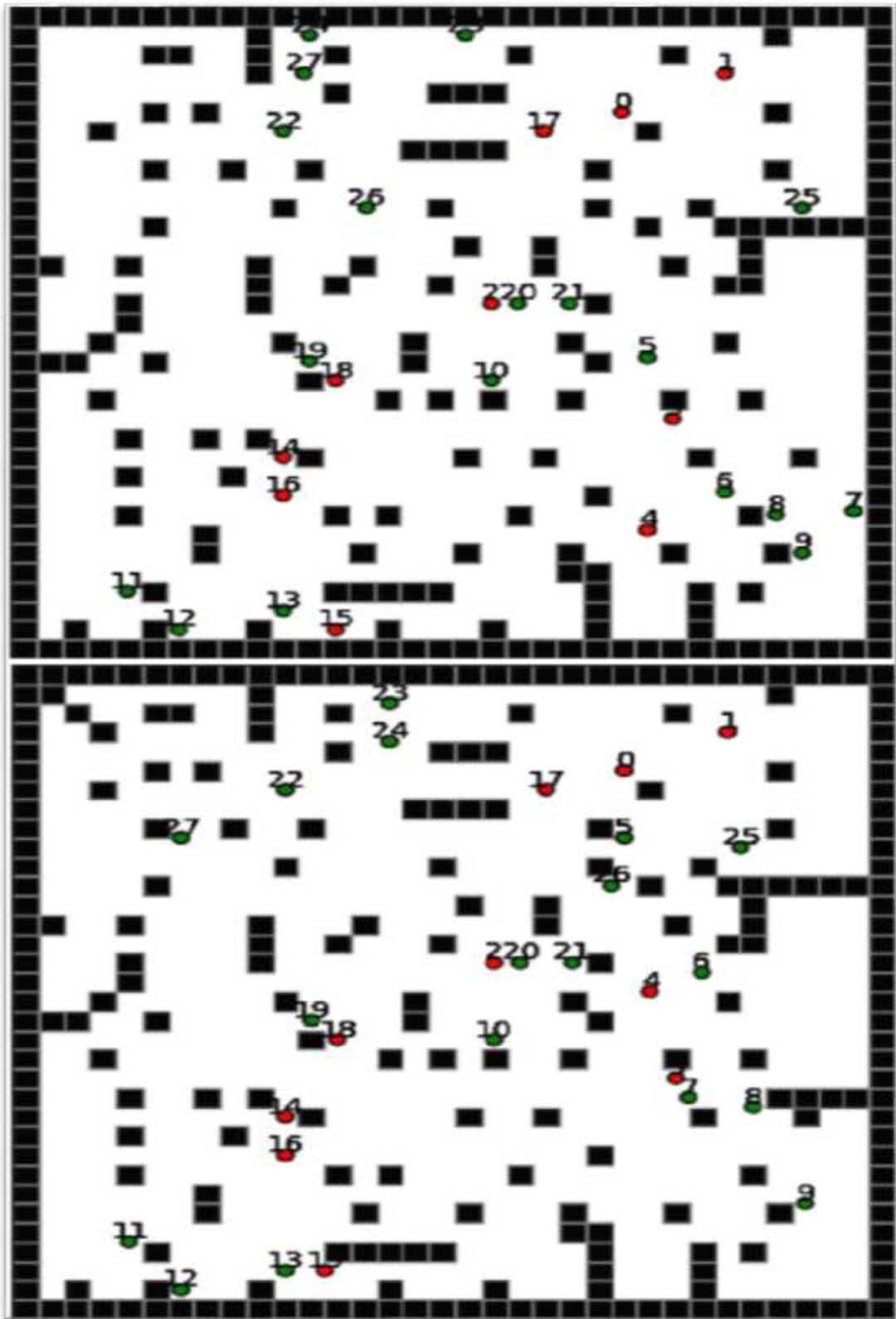


Figure 4.13: Shows Another Example of The Second Scenario.

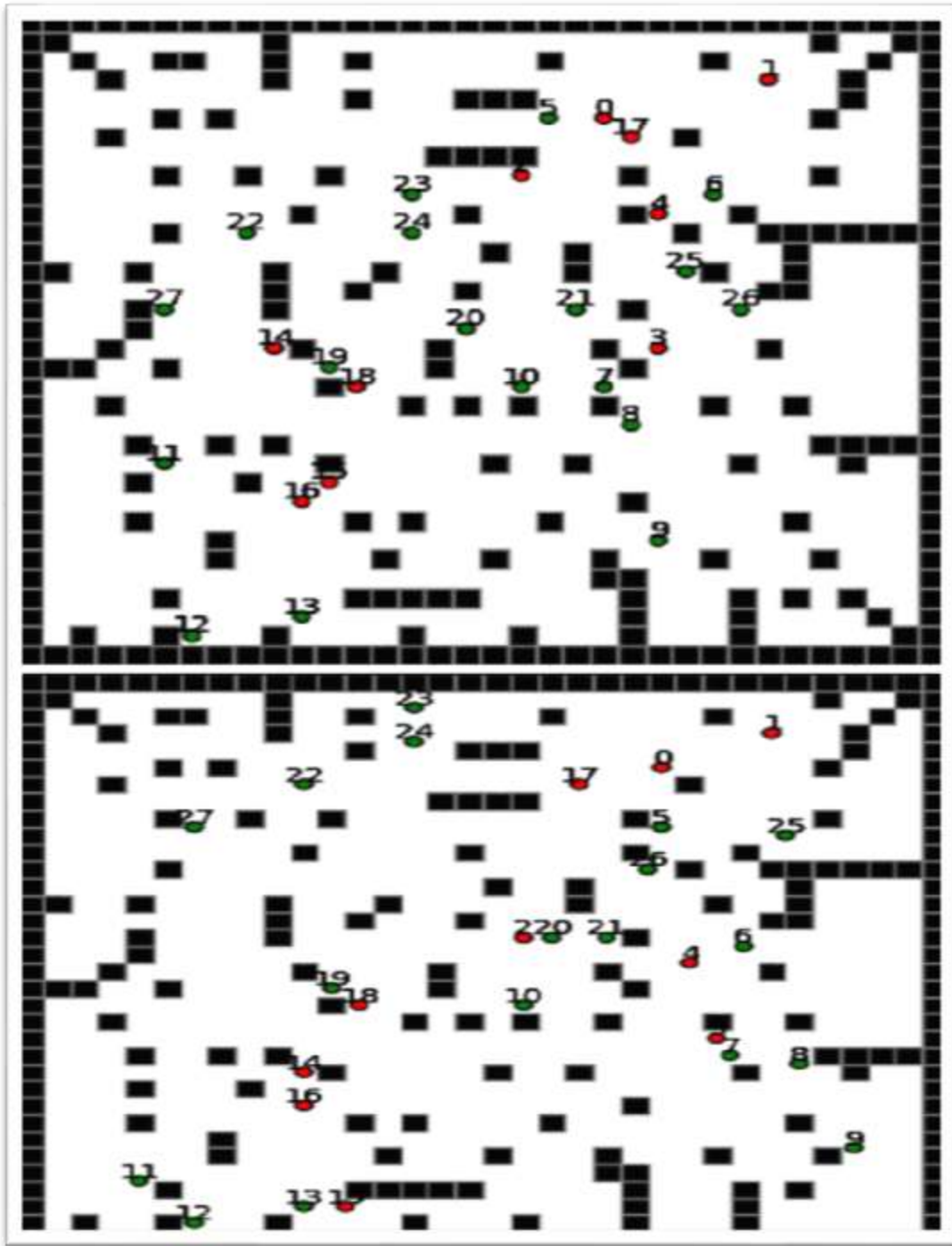


Figure 4.14: Shows Example of Increasing the Complexity of The Environment.

Figure 4.14 shows us the presence of several obstacles of different shapes to increase the complexity of the environment. The result also shows us the process of avoiding fixed and moving obstacles by the angels in order to reach the goals.

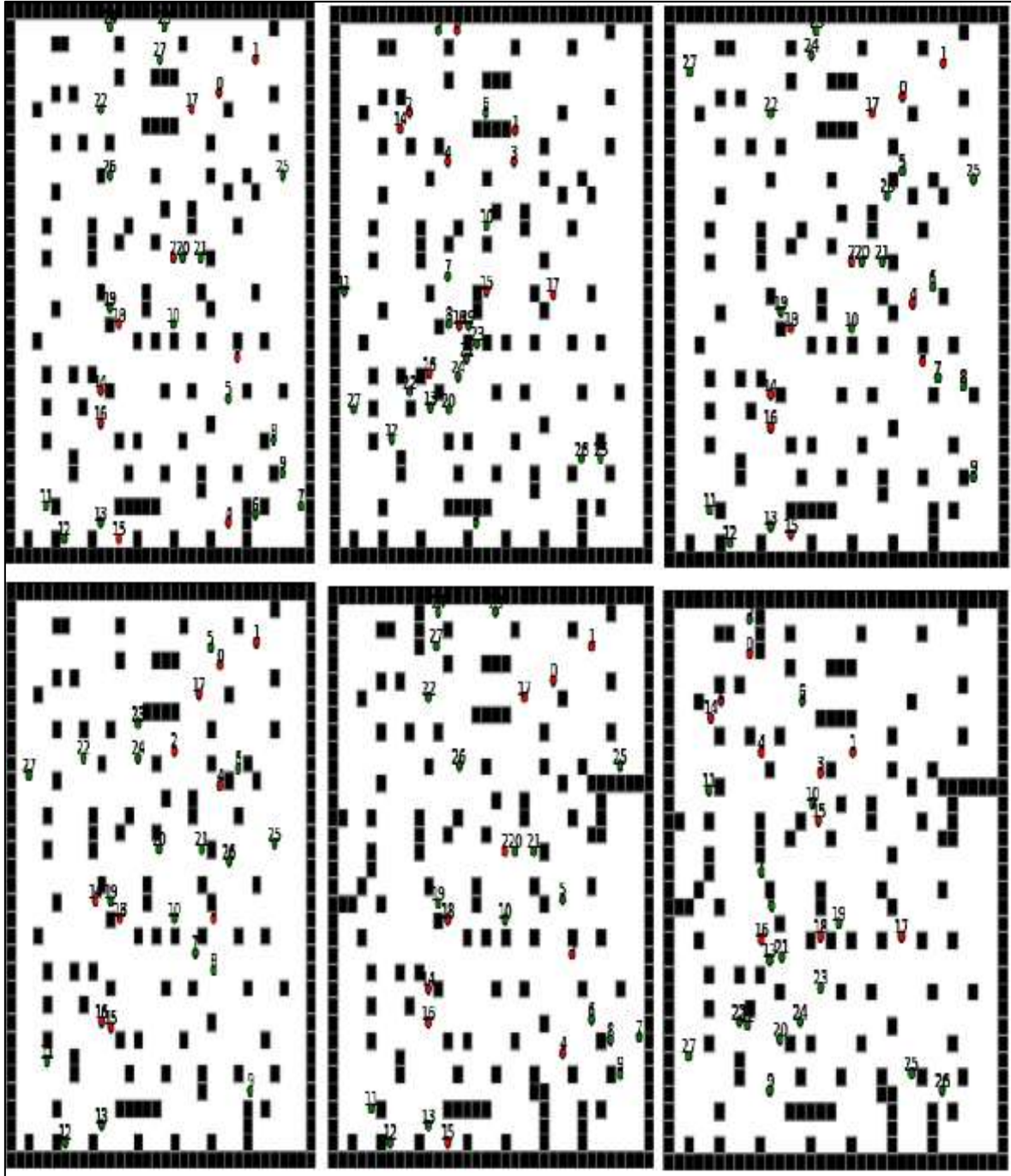
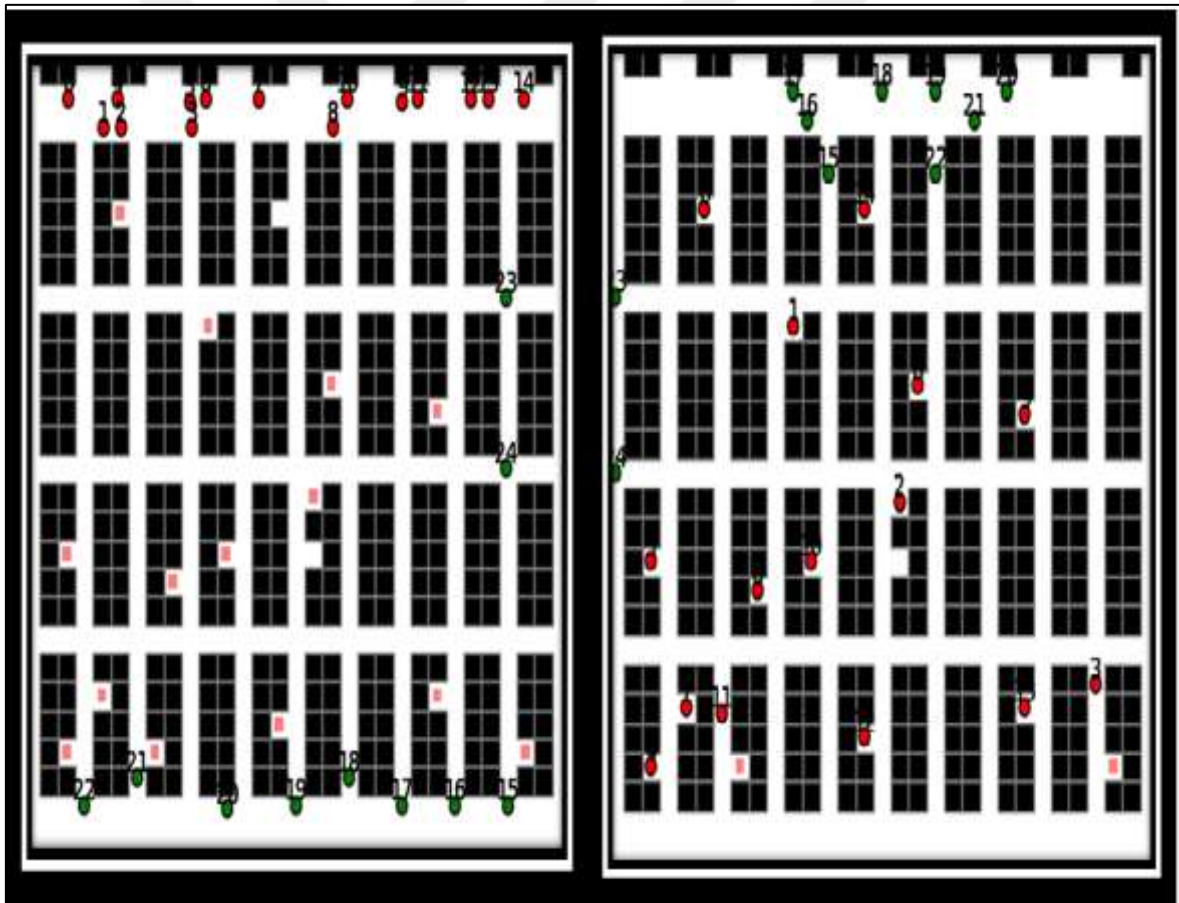


Figure 4.15: Illustrates The General Path of The Second Scenario.

Figure 4.15 shows several examples of multiple environments of varying complexity to test agents in the most difficult environments and try to reach their targets smoothly without colliding with any obstacle, whether fixed or moving, within the environment.

4.2.3 Third Scenario in A Warehouse Environment

One of the most talked-about issues in warehousing recently has been multi-targeting [167]. It is very important that in an industrial environment-like large-scale warehouses-agents can respond and navigate as quickly as possible. Several simulation tests and illustrates various industrial warehouse scenarios in which to prove the efficacy of our approach. Having defined the starting and destination positions of each agent, the objective will be that of providing each robot with a trajectory between those points efficiently, while providing all the clear paths devoid of obstacles and collisions free. We have designed several scenarios that shall serve to exemplify our method's capabilities by underlining effectiveness in the management of complex multi-agent environments and ensuring safety, collision-free navigation across varied warehouse configurations [167].

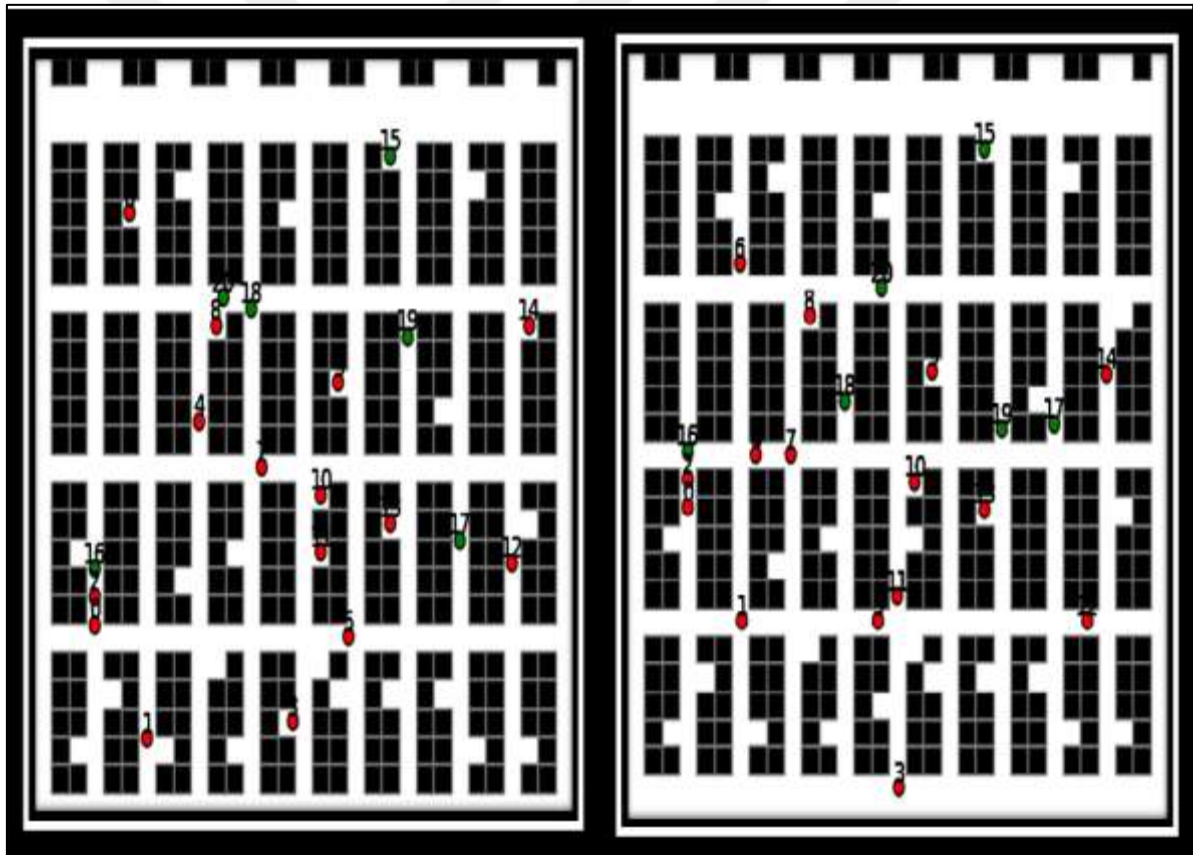


(a) Multi-agent start position.

(b) Multi-agent reaches the first target.

Figure 4.16: The Proposed Experiment Works First Target Setting.

An initial configuration of points and targets of the warehouse is shown in Figure 4.16(a). In this case, the robot moves skillfully towards its destination, avoiding obstacles moving through its path. When obstacles keep flowing through, it can still avoid collision and reach its destination as shown in Figure 4.16(b). After reaching its first target, the status of the robot changes and new targets are positioned in the environment. Now, the robot uses the target location it reached earlier as a new starting point. The robot has to replan a route in order to accommodate the newer challenges such as increase/decrease in a number of obstacles and changes in the location of goals. This dynamic adaptation is relevant for the assessment performance of a proposed system in different scenarios, hence allowing the algorithm to be effective under ever-changing conditions and to continue its task of efficient and collision-free navigation.



(a) New start state.

(b) Multi-agent reaches the second target.

Figure 4.17: New Targets Have Been Assigned.

Figure 4.17 (a) The robot starts from an objective that has already been reached and becomes its new starting point. After the updates with the new target positions, the robot recalculates

its route toward those updated targets. Remarkably, moving obstacles increase in lower number; hence, a new phase of changes is introduced: the robot is assigned new targets. In that respect, the robot needs to realign its priorities and replan its travel through the revised environment complexity. Figure 4.17(b) shows two phases of goal changes that the robot has gone through. First, having passed the first target, the status of the robot was updated, and new starting points were set, as well as new goals. Thus, in this scenario, the environment changed twice, being a continuously changing environment; this algorithm is capable of handling those through continuous recalculations and replanning based on the changes of the environment. Once it achieved the initial goal, the information the robot originally possessed was updated to reflect new starting points. New goal points were created to consecutively modify the environment. The succeeding table presents those changes: how the environment was modified and how the new goals were added into the system. This shows just how dynamic the scenario is and how the algorithm dynamically adapts to the change in conditions.

Table 4.3: Describe The New Goal Position That Have Been Occurred to The Mobile Robots.

First Start	First Goal	Second Start	Second Goal	Moving obstacle	Map description
(0, 0)	(5,5)	(5,5)	(24,2)	28	48*48
(0 ,3)	(9 ,10)	(9,10)	(24,7)	28	48*48
(0, 4)	(15 ,16)	(15,16)	(22,4)	28	48*48
(0 ,8)	(24, 28)	(24,28)	(23,14)	28	48*48
(0, 7)	(24, 2)	(24,2)	(5,14)	28	48*48
(0 ,11)	(17 ,2)	(17,2)	(24,28)	28	48*48
(0, 12)	(18, 8)	(18,8)	(5,5)	28	48*48
(0, 15)	(22, 4)	(22,4)	(12,23)	28	48*48
(0 ,16)	(11, 17)	(11,17)	(9,10)	28	48*48
(0, 19)	(12, 23)	(12,23)	(11,17)	28	48*48
(0, 20)	(17, 11)	(17,11)	(15,16)	28	48*48
(0 ,24)	(24, 7)	(24,7)	(17,16)	28	48*48
(0 ,27)	(23, 14)	(23,14)	(16,28)	28	48*48
(0 ,28)	(22, 23)	(22,23)	(16,20)	28	48*48
(0 ,30)	(5 ,14)	(5,14)	(9,28)	28	48*48
(28, 28)	(1 ,2)	(1,2)	(3,20)	28	48*48
(28,25)	(1,7)	(1,7)	(27,2)	28	48*48
(28,22)	(1,10)	(1,10)	(27,23)	28	48*48
(28,18)	(1,15)	(1,15)	(1,12)	28	48*48
(28,14)	(1,18)	(1,18)	(1,20)	28	48*48
(28,10)	(1,22)	(1,22)	(8,0)	28	48*48
(28,6)	(1,25)	(1,25)	(24,2)	28	48*48
(28,2)	(1,28)	(1,28)	(24,7)	28	48*48
(8,30)	(8,0)	(8,0)	(22,4)	28	48*48

Table 4.3 shows the transformation that has taken place by the robots doing their job. The third column of this table, which was based on the second column, now represents new starting points for the robots. This update mechanism ensures that, upon changes in the map or introduction of new points, both the starting and destination points get updated based on the preceding ones in the sequence. This dynamic updating is particularly challenging where such environments involve not only fixed obstacles, such as configurations of storage, but also moving obstacles in the form of humans or warehouse workers.

The Table 4.4 below shows how such adjustments are managed to maintain the path planning efficiently and precisely under such complex and dynamic conditions. After the initial set of goals has been refined, replanning of the routes is next. When the robots reach their now updated goals, this set of goals is refined for a second time. Further revisions will, therefore, result from the new starting positions, which will be computed from the previously defined points. Here, in this experiment, we work in an environment enveloped by a certain number of moving obstacles that are increased or decreased by the particular parameters of the environment. Putting it into the example of a warehouse, this can be about expanding the number of objectives to be sorted, adding more personnel, or even relocating secret safes. These are dynamic changes that involve continuous reconsideration and re-planning in light of effective pathfinding and navigation. While developing the strategy, one needs to consider all the eventualities and changes that take place in the environment. Having addressed these considerations, we now move on to present to you the final outcome of our work. This outcome, inclusive of implications of our strategy and its effectiveness across various scenarios, will be shown in the Figure 4.18:



Figure 4.18: The Final Destination Have Been Reached By The Mobile Robot After Pick and Delivery Operation.

The resulting figure will give a complete picture since it will show how the strategy performs under different scenarios, considering how appropriate the approach was in allowing both static and dynamic obstacles. First, the final destination of the robot was its original starting point. However, by modifications in the environment, the robot divided the goals into two sub-goals. The completion of those sub-goals brought the robot back to the original starting position. Simulation contained multi mobile robots, 520 static obstacles, and moving human workers.

Table 4.4: The Results Describe the Warehouse Environment and The Change That Occurred Three Times in It.

Start state	Goal state	time	Path length	Moving obstacle	Map description
(24,2)	(0, 0)	6.4	8.9	28	48*48
(24,7)	(0 ,3)	5.3	6.4	28	48*48
(22,4)	(0, 4)	4.2	9.5	28	48*48
(23,14)	(0 ,8)	3.4	7.3	28	48*48
(5,14)	(0, 7)	9.4	6.5	28	48*48
(24,28)	(0 ,11)	8.7	5.5	28	48*48
(5,5)	(0, 12)	6.9	6.2	28	48*48
(12,23)	(0, 15)	5.4	3.7	28	48*48
(9,10)	(0 ,16)	4.6	3.2	28	48*48
(11,17)	(0, 19)	7.6	3.5	28	48*48
(15,16)	(0, 20)	4.9	4.4	28	48*48
(17,16)	(0 ,24)	7.6	3.2	28	48*48
(16,28)	(0 ,27)	6.5	4.3	28	48*48
(16,20)	(0 ,28)	7.6	5.4	28	48*48
(9,28)	(0 ,30)	8.2	3.2	28	48*48
(3,20)	(28, 28)	6.2	5.3	28	48*48
(27,2)	(28,25)	4.3	6.2	28	48*48
(27,23)	(28,22)	5.2	6.5	28	48*48
(1,12)	(28,18)	3.5	4.4	28	48*48
(1,20)	(28,14)	5.3	3.3	28	48*48
(8,0)	(28,10)	4.6	6.1	28	48*48
(24,2)	(28,6)	6.2	7.2	28	48*48
(24,7)	(28,2)	5.1	8.6	28	48*48
(22,4)	(8,30)	3.4	7.2	28	48*48
(24,2)	(0, 0)	3.8	11.1	28	48*48
(24,7)	(0 ,3)	7.6	9.2	28	48*48
(22,4)	(0, 4)	2.4	8.4	28	48*48
(23,14)	(0 ,8)	6.4	5.6	28	48*48
(5,14)	(0, 7)	4.5	4.4	28	48*48

Table 4.4 presents the current status of the robot targets. In relation to Table 4.4, through the analysis of the departure and arrival points, it is possible to gauge changes in all the stages of the trajectory travelled by the robots. It also makes clear in this comparison the main and secondary objectives that have surfaced during all the simulation. Table 4.5, below,

effectively illustrates how those initial objectives were refined to reflect changes in the dynamic environment and the consequences of such changes on routing and task management by robots. The strategy has shown very effective control over the mapped environments, maintaining a very high percentage of success within the simulation. Figure 4.22 illustrate the performance of the robots for various scenarios. Each robot is initially reset, then robotically navigates its path through dynamic obstacles to reach its first goal and then on to a newly set goal and back again to the workstation. Success in all these configurations is an indication of the robustness of the proposed strategy against adaptation to new, multiple targets or dynamic challenges.

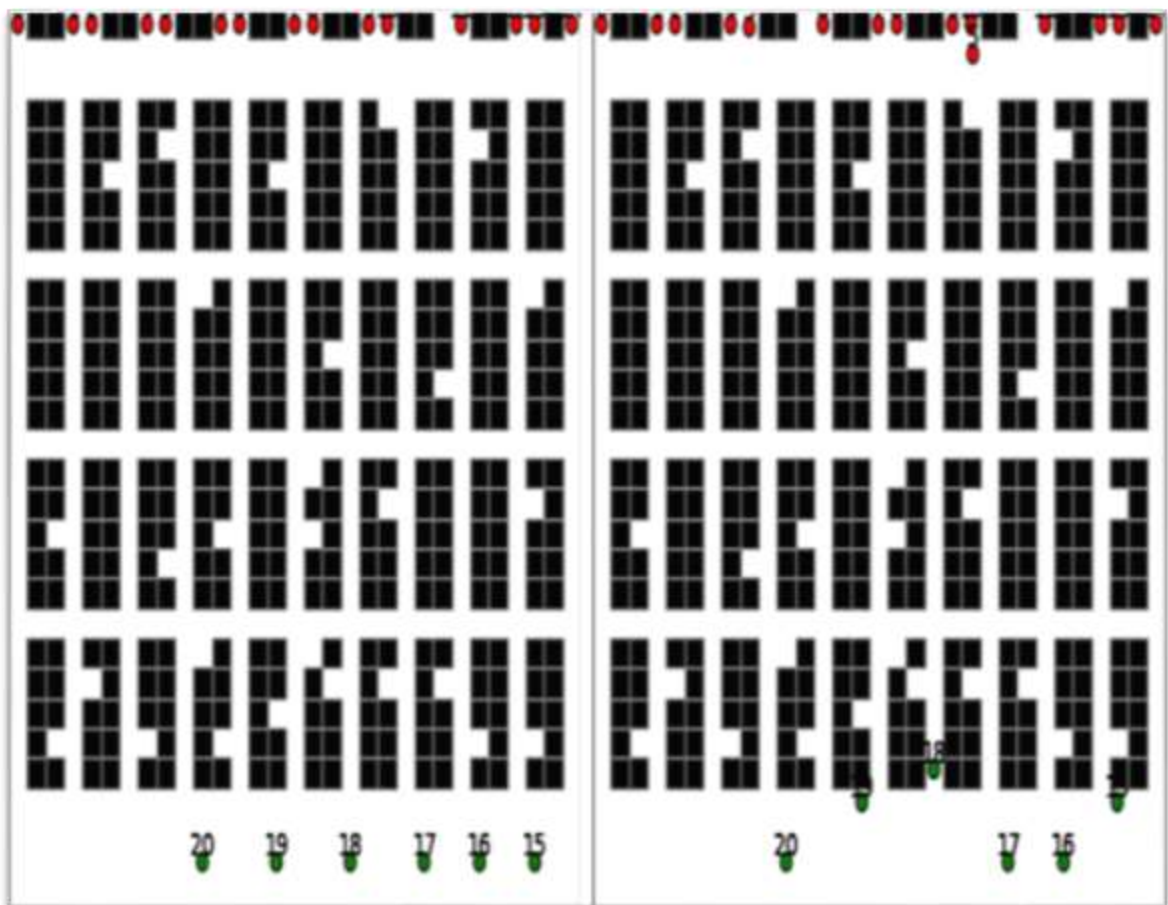


Figure 4.19: Shows The Start State, Goal, And the Obstacle Position.

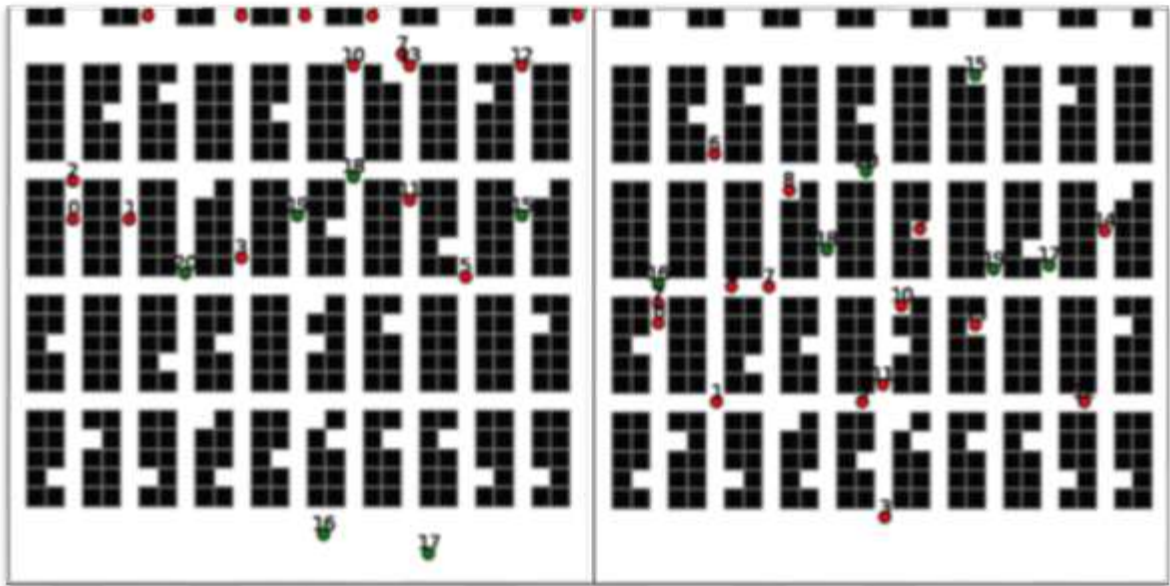


Figure 4.20: Shows The Environment with The Moving Obstacle.

Figure 4.20, shows us the process of each customer moving towards their goals and they have to pass through fixed and moving obstacles to reach them. The next result will show the second target setting.

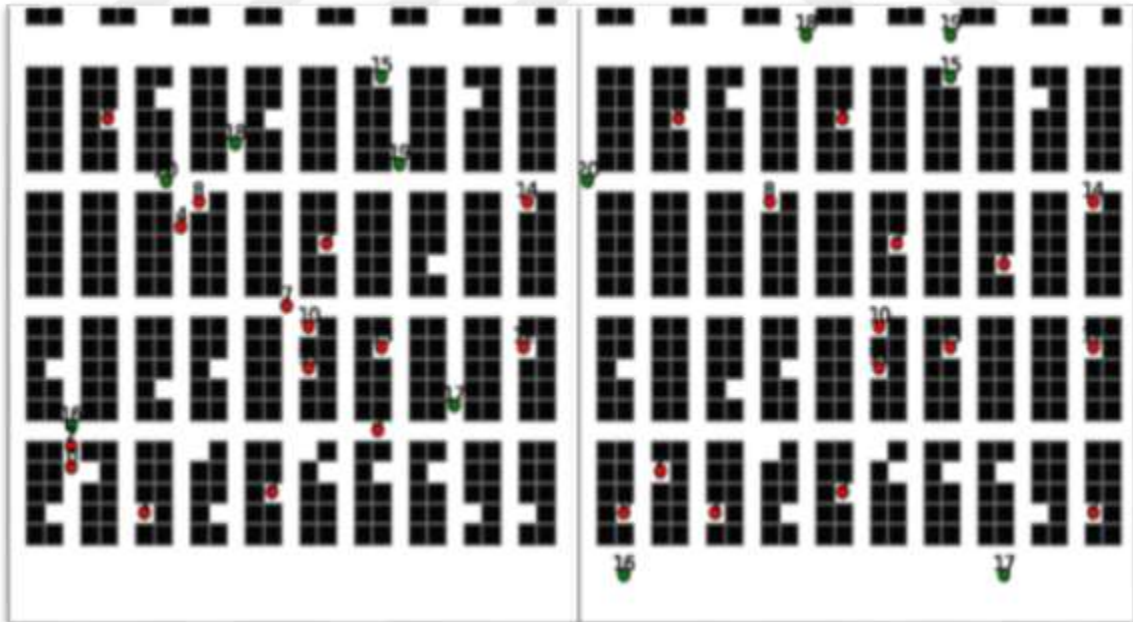


Figure 4.21: Changing In the Environment Example.

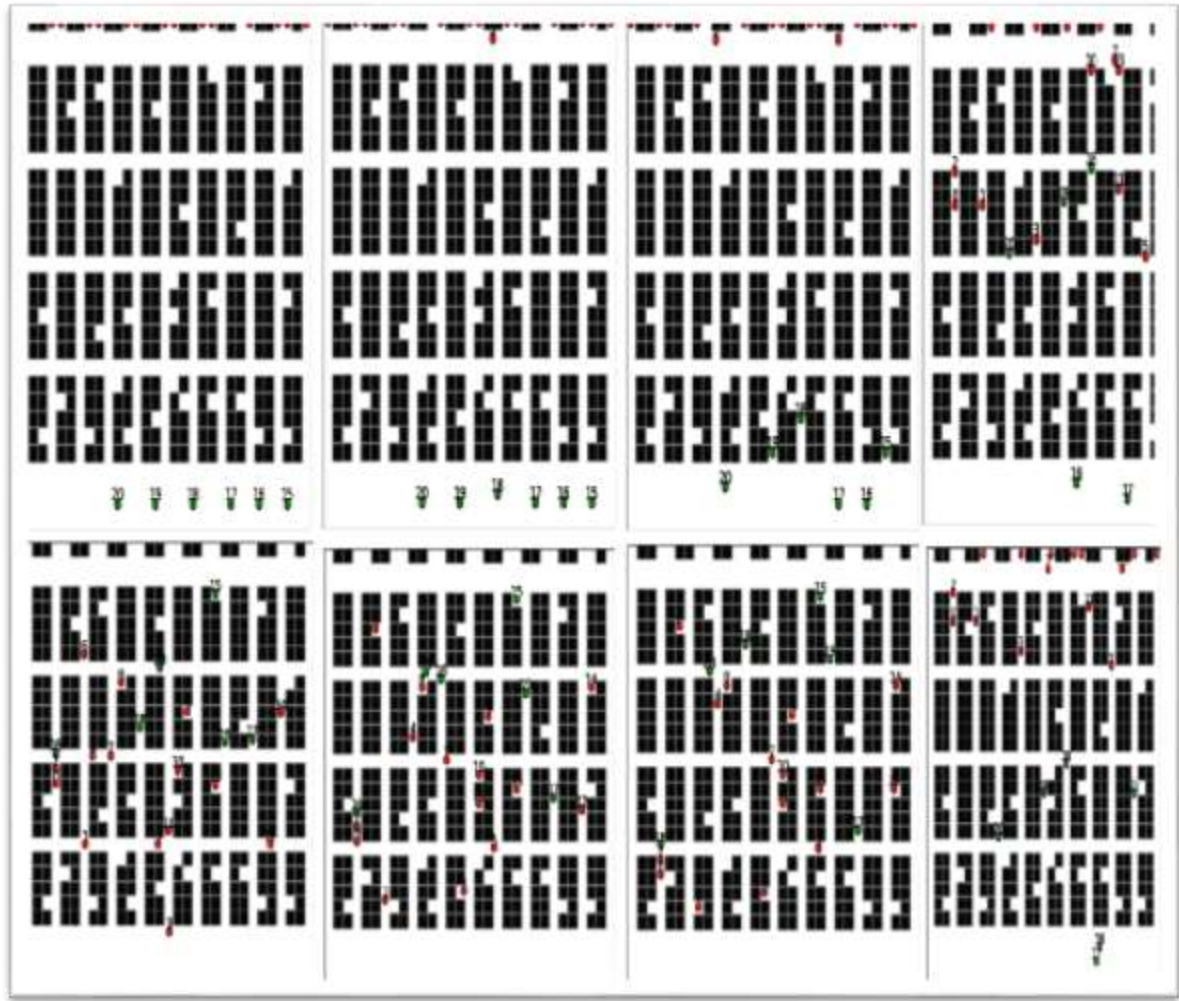


Figure 4.22: Overall System Experiment.

Figure 4.22 summarizes all the changes to the experiments and shows how the strategies adapt to changes and complications in these above-mentioned environmental changes. This high percentage of success rate emphasizes the fact that the approach is pretty robust in operation and handling of complex scenes in dynamics. The following table represents a comparison of all environmental changes based on both time and the length of the journey. This table also explains target assignment shifts that have taken place in the warehouse context. It points out the results of three different goal assignments: agents moving away from their home locations constitute the first goal, and then new targets were assigned to the agents in the second goal, and further changes were made to those targets in the third goal. The fourth column in Table 4.5 was the modification of the pathways to show how the change in pathways affects the performance of the multi-agent system. The above analysis

in detail will provide insight into how each step-target assignment and alteration of the environment contribute toward overall system efficiency and performance.

Table 4.5: Result and Comparison Between All the Changes That Occurred in The Environment.

First goal result (Sec)	Second goal result (Sec)	Third goal result (Sec)	First Path Change (T in Sec)	Second Path Change (T in Sec)	Third Path Change (T in Sec)
7.5	5.5	6.4	7.9	6.7	8.9
3.9	4.9	5.3	5.4	4.9	6.4
9.3	8.3	4.2	12.5	10.1	9.5
10.1	9.8	3.4	6.3	5.2	7.3
7.7	5.6	9.4	5.5	4.3	6.5
6.4	6.4	8.7	6.3	6.3	5.5
8.6	9.6	6.9	4.2	5.1	6.2
5.9	4.6	5.4	6.7	4.5	3.7
2.6	3.6	4.6	4.2	3.1	3.2
4.2	5.2	7.6	4.5	3.9	3.5
3.3	3.8	4.9	5.4	3.9	4.4
2.5	2.1	7.6	4.2	4.9	3.2
7.9	5.6	6.5	5.3	6.2	4.3
5.2	4.2	7.6	3.1	5.2	5.4
4.0	4.9	8.2	2.2	4.1	3.2
3.4	3.3	6.2	4.3	6.3	5.3
9.5	6.5	4.3	5.2	6.7	6.2
4.5	5.5	5.2	4.5	7.5	6.5
3.2	4.2	3.5	3.4	4.1	4.4
2.2	1.9	5.3	6.1	4.2	3.3
7.6	6.5	4.6	7.1	8.1	6.1
6.9	7.1	6.2	6.2	9.2	7.2
10.5	5.2	5.1	10.6	7.6	8.6
9.3	6.1	3.4	9.2	11.2	7.2
8.1	7.2	3.8	12.1	8.1	11.1
7.6	6.2	7.6	8.2	6.2	9.2
6.8	5.9	2.4	7.4	5.4	8.4
4.6	5.4	6.4	6.6	5.6	5.6
3.5	4.9	4.5	3.4	2.4	4.4

Table 4.5 presents a thorough analysis of the consequences of altering goals at three phases. It emphasizes the influence on diverse outcomes, focusing on a 48x48 space with several static and dynamic impediments. This table highlights the performance disparities noticed in response to goal adjustments. The following results will focus on the success rate and efficacy of avoiding barriers, particularly in situations including the addition, removal, and alteration of impediments.

5. DISCUSSION

Figures 5.1 and 5.2 show how well the system handles numerous obstacle modifications, such as varying the number of barriers and generating random maps. The suggested technique is versatile in managing several types of random maps, each with its own level of complexity, which might impact the path planning process. Based on testing in a variety of unfamiliar locations, the study suggests that the suggested technique efficiently addresses the problems of path planning.

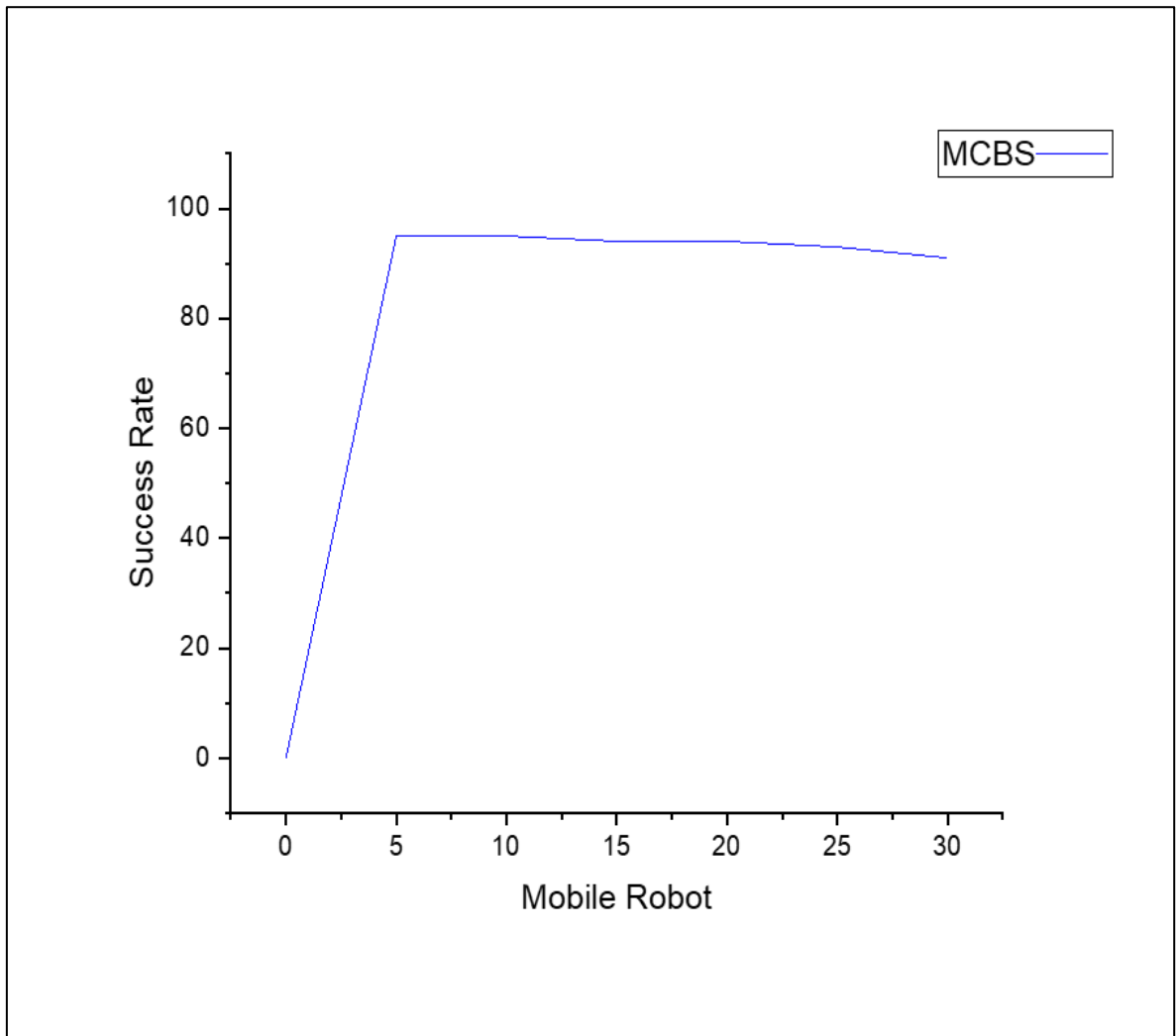


Figure 5.1: Shows The Success Rate of Our Modified Algorithm Especially When Avoiding Obstacles in Path Trajectory.

Figure 5.1 shows the success rate of customers in reaching the goals. It also shows the extent of the success of the improvements that were added to the proposed work.

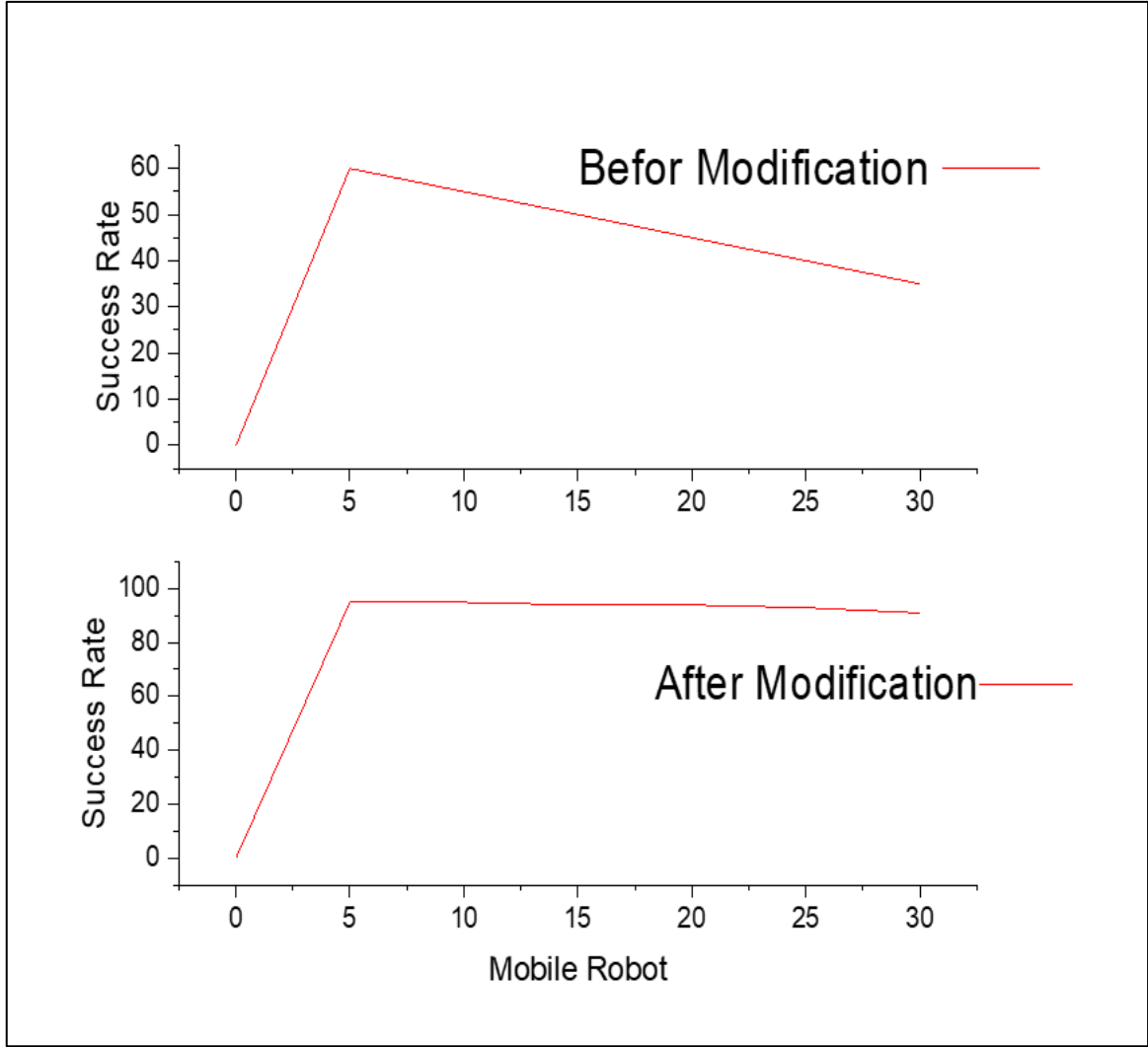


Figure 5.2: Shows The Success Rate of Avoiding Obstacles in Path Trajectory. These Results Show Before and After the Modifications and Improvements That Were Made.

Figures 5.1, 5.2, and 5.3 depict the whole obstacle avoidance procedure as well as the updated planning method's capacity to manage transitions between single and multi-target scenarios. The success rate assessment evaluates the effectiveness of the suggested strategy in dynamic contexts. Environmental changes might have an influence on an agent's ability to achieve their aims. For example, in a warehouse scenario, while the original objective setting may have a little effect, later adjustments such as establishing a second goal might raise complexity, priority considerations, and expenses, which may indirectly influence the success rate.

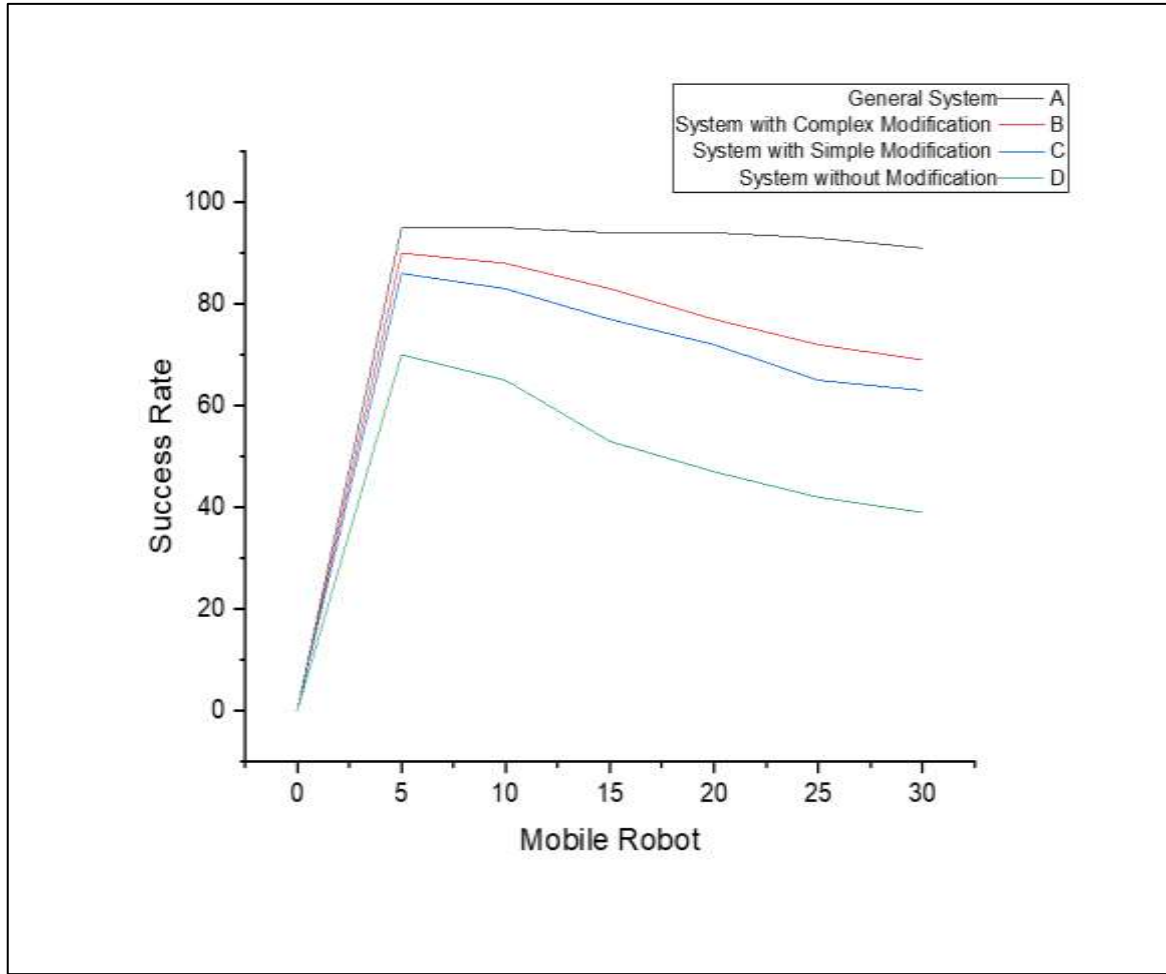


Figure 5.3: Shows The Number Average of Adding the Modification to The System.

Figure 5.3 shows, respectively, four examples of how the proposed work has been modified. In the worst case, only minor adjustments have been performed. In the second one, tasks were assigned within the complex environment; slight modification of the overall work is shown in result due to the distribution of the loads. The third one showed that as the number of robots in the environment increases, so does the complexity. That complexity depends on the difficulty of planning for the achievement of the goal and the degree of changes within this environment. Finally, in the general system, there was a load distribution rate introduced with an absolute priority regarding the issue of path shortening. It was achieved by measuring the distance between the goals and then optimizing the route for the shortest access to each goal.

The success rate in MAPF refers to the proportion among all problem instances where there is a valid, conflict-free solution for all agents [137-146]. A solution is considered to be

successful if all of them reach their respective goals without collisions, normally within a certain amount of time or computational limit. It is probably the most important indicator of the performance of the MAPF algorithm, especially in highly dense agent environments or complex configurations of obstacles. Since the underlying problem becomes increasingly hard with an increased number of agents, map complexity, or number of obstacles, the probability of finding a valid solution decreases, therefore reducing the success rate. Several factors are directly related to the success rates in MAPF. One such factor is the density of agents, which plays a vital role in that with the increase in the number of agents about space, the conflicts also tend to increase in such a way that it becomes tough to find a feasible path for all agents as mentioned in[138]. In settings that are dense, agents are more likely to meet each other, hence decreasing the success rate as an algorithm tries to resolve them. The complexity of the grid or map also alters the success rate. Maps with a lot of obstacles, thin corridors, or limited movement options drive agents into closer proximity, increasing the difficulty to avoid conflicts. Whereas open maps with fewer obstacles would result in a higher success rate due to the flexibility for agent movements, on the other hand.

The efficiency of the algorithm itself also contributes much to the success rate. Some algorithms, like Conflict-Based Search (CBS), are complete and optimal; that is, if there exists a solution, they find it [1]. However, this might degrade with computational limitations in large-scale or complex scenarios, thus having a lower success rate in these scenarios. Heuristic or approximate algorithms could provide faster solutions at the cost of optimality, by which they are not able to solve all the conflicts and hence always yield fewer successful cases. Success rate also depends on the priorities used for resolving agent-agent conflicts. While it simplifies the task of resolving conflicts, assignment of poor priorities results in suboptimal solutions or "starvation" of some agents. This happens when lower-priority agents rarely find a valid path, hence generally decreasing the success rate.

The second important issue concerns the time limit or computational constraints of the algorithm. Often, in field applications, algorithms need to yield a solution within some bounded execution time. If the solution is not found in time, then the instance is tagged as a failure, which reduces the success rate. This will be very relevant in real-time applications where timely decisions are crucial. Equally important for an algorithm's efficiency will be replanning paths in dynamic environments, wherein obstacles or agents might change across

the course of a simulation. A setting like this will be capable of retaining an increased success rate for algorithms with strong real-time replanning capabilities, while the system that is unable to adapt quickly will tend to reduce its success rate with any changes in conditions. Success rate becomes an overall crucial factor that can explain how well an MAPF algorithm will perform in the case of a variety of tested problem instances of different complexities. A high success rate testifies that the algorithm is able to effectively solve conflicts and find valid paths even in tricky environments, while lower values might indicate either that the algorithm is not effective against particular configurations, or it is at least bounded by computational resources.

Various pathfinding algorithms, such as PRM, RRT, A*, D*, Dijkstra, GA, ACO, PSO, VO, and QL, have their respective pros, cons, and limitations concerning the environment, complexity, and the number of agents in the MAPF domain. The following Table 5.1 shall discuss these algorithms in detail.

Table 5.1: The Key Advantage of The Related Method.

Type of method	Environment	Limitation	Advantage	Disadvantage
PRM	Mixed	Difficulty in dynamic environments and not optimal for dense agent scenarios	Scalability, Preprocessing in known environment	Quality of paths and No guarantees on completeness
RRT	mixed	Conflict resolution and not ideal for dynamic environments	Exploration of complex environments	Path quality and requires post-processing
A*	Mixed	Handling of dynamic obstacles and Conflict resolution between agents	Heuristic guidance	Scalability and memory usage
Dijkstra	Mixed	Poor scalability and Handling of dynamic obstacles	Simplicity	Inefficiency
GA	Mixed	Coordination between agents	Flexibility	Slow convergence and Randomness
ACO	Mixed	Conflict resolution and Scalability	Adaptability	Slow convergence and Sensitivity to parameter tuning
PSO	Mixed	Conflict resolution and Convergence issues	Adaptability	Lack of guaranteed optimality and parameter issue
VD	Mixed	Scalability issues and not optimal	Simplicity	Local behaviour and Over-conservatism
QL	Mixed	Scalability	Learning-based	Long training times and exploitation dilemma

CBS has many advantages compared to most other Multi-Agent Path Finding algorithms, but again, whether it is "better" depends on the scenario. CBS is designed with MAPF specifically in mind and as such inherently picks up strong properties of multi-agent coordination and conflict resolution. It solves conflicts incrementally and its key feature allows to find optimal collision-free paths for all agents while maintaining completeness [167]. This also makes CBS particularly effective in mixed environments, where agents need to avoid collisions from dynamic changes.

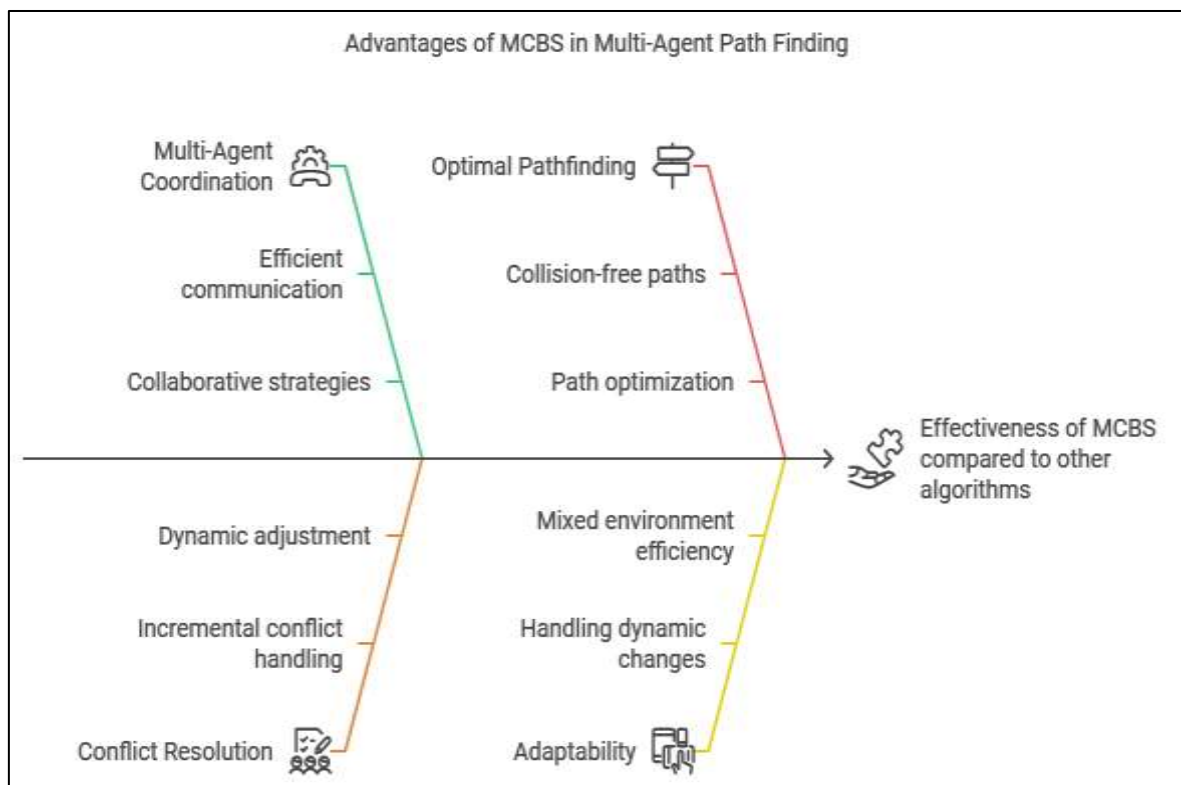


Figure 5.4: Modified CBS Algorithm Overview.

Compared to algorithms such as A* and Dijkstra, multi-agent pathfinding is handled more efficiently with CBS, since these single-agent algorithms must be augmented with additional conflict resolution methods when used for MAPF. While optimal in the case of single-agent pathfinding, neither A* nor Dijkstra natively address multi-agent interactions. CBS does, offering a more formalized approach to the avoidance of conflicts.

PRM and RRT are indeed effective heuristics for exploring a large, complicated space; however, they do not easily deliver optimality or guarantees of conflict-free solutions. The sampling-based algorithms would probably behave even better in high-dimensional spaces

or very large environments; however, generally speaking, CBS can be trusted more for multi-agent coordination in usual grid or structured spaces.

While heuristics such as GA, ACO, and PSO can be flexible and find solutions in large or complex environments, they give up optimality to suffer from slow convergence or inconsistency. Meanwhile, CBS is optimal and complete; it is more reliable in scenarios where finding the best possible path is really crucial. In conclusion, CBS is better with respect to MAPF.

This impact is further observed with additional target settings, reflecting how the method adapts to evolving challenges. In particular, a comparison was made of the proposed method with relevant prior work to validate its high performance as shown in Table 5.2.

Table 5.2: The Comparison of Modified CBS Result with Mentioned Related Works.

Author	Type of method	T sec	Average length	MAP	Robot type	Obstacle number	Obstacle type
[141]	Soft Actor-Critic	185.6	55.3	Static	Ship-Car	4	Concave
[164]	DWA	169.8	34.5	Dynamic	Car-like	6	Mixed
[155]	Ds	177.5	45.3	Dynamic	Car-like	Random	Concave
[142]	K-PBS	248.7	NA	Dynamic	Car-like	Random	Concave
[153]	SBAAs	NA	NA	Dynamic	Omni-Direction	Random	Concave
[138]	CBS	165.4	64.2	Dynamic	Omni-Direction	Random	Concave
[166]	GNN	NA	NA	Static	NA	Random	Concave
Our	MCBS	100.4	32.1	Static, Dynamic	Omni-Direction	Random up to 50	Mixed

6. CONCLUSION

This research was conducted in several virtual environments to test performance and improve the process of achieving goals. The study began with tests in a simple environment, gradually increasing complexity by adding new obstacles. Implementing the algorithm in both static and dynamic environments posed challenges in defining a clear and precise path. However, the experiments demonstrated the efficiency of the approach. Additionally, modifications were made to the course, and multiple goals were selected to enhance complexity and evaluate the validity of the hypotheses. Analysis of the results, compared with initial findings, confirmed that the study met its objectives. The findings indicate that this work can effectively reach multiple goals, return to a primary base, and avoid both fixed and moving obstacles within the environment. The study addresses the challenge of obstacle avoidance in multi-target systems within unknown environments. The main challenge involved implementing mobile robot path planning for multiple targets while navigating unknown settings. In situations without restrictions on the routes, robots can avoid colliding with other agents. This thesis tackled this challenge using a modified Conflict-Based Search (CBS) algorithm. Adjustments made to the agents significantly improved performance, particularly in warehouse applications. The research meets the criteria for various path planning issues, effectively addressing static, dynamic, unknown, and warehouse environments. Several enhancements were implemented in both the environment and the algorithm regarding shortest path and obstacle avoidance capabilities. The proposed approach demonstrated high performance in terms of success rate, time efficiency, average path length, and trajectory for multiple targets. This research utilized multiple agents, and future work aims to explore different criteria, such as 3D path planning and applications in industrial environments.

REFERENCES

- [1] Zhang, Lixiang, et al. "Multi-agent policy learning-based path planning for autonomous mobile robots." *Engineering Applications of Artificial Intelligence* 129 (2024): 107631.
- [2] Mao, Jianlin, et al. "Multi-Agent Collaborative Path Planning Algorithm with Multiple Meeting Points." *Electronics* 13.16 (2024): 3347.
- [3] Lee, Minjae, et al. "A proposal on multi-agent static path planning strategy for minimizing radiation dose." *Nuclear Engineering and Technology* 56.1 (2024): 92-99.
- [4] Lu, Mai-Kao, et al. "Hierarchical Q-Learning Path Planning for Cooperative Tracking Control of Multi-Agent Systems With Lumped Uncertainties." *IEEE Transactions on Automation Science and Engineering* (2024).
- [5] Su, Yifan, Rishi Veerapaneni, and Jiaoyang Li. "Bidirectional Temporal Plan Graph: Enabling Switchable Passing Orders for More Efficient Multi-Agent Path Finding Plan Execution." *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 38. No. 16. 2024.
- [6] Yan, Jingtian, and Jiaoyang Li. "Multi-Agent Motion Planning with Bézier Curve Optimization under Kinodynamic Constraints." *IEEE Robotics and Automation Letters* (2024).
- [7] Chen, Zhe, et al. "Traffic flow optimisation for lifelong multi-agent path finding." *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 38. No. 18. 2024.
- [8] Kottinger, Justin, et al. "Introducing Delays in Multi Agent Path Finding." *Proceedings of the International Symposium on Combinatorial Search*. Vol. 17. 2024.
- [9] Phan, Thomy, et al. "Adaptive Anytime Multi-Agent Path Finding Using Bandit-Based Large Neighborhood Search." *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 38. No. 16. 2024.
- [10] Chen, Weizhe, Sven Koenig, and Bistra Dilkina. "Why solving multi-agent path finding with large language model has not succeeded yet." *arXiv preprint arXiv:2401.03630* (2024).

- [11] Ni, Xinrui, et al. "A Q-learning based multi-strategy integrated artificial bee colony algorithm with application in unmanned vehicle path planning." *Expert Systems with Applications* 236 (2024): 121303.
- [12] Zhang, Yulun, et al. "Guidance Graph Optimization for Lifelong Multi-Agent Path Finding." *arXiv preprint arXiv:2402.01446* (2024).
- [13] Phan, Thomy, et al. "Anytime Multi-Agent Path Finding with an Adaptive Delay-Based Heuristic." *arXiv preprint arXiv:2408.02960* (2024).
- [14] WU, Wenjun, et al. "Survey of Multi-agent Path Finding Technology." *Journal of Beijing University of Technology*.2024.
- [15] Puente-Castro, Alejandro, et al. "Q-learning based system for path planning with unmanned aerial vehicles swarms in obstacle environments." *Expert Systems with Applications* 235 (2024): 121240.
- [16] Zhang, Dengxing, Chen Chen, and Guanyu Zhang. "AGV path planning based on improved A-star algorithm." *2024 IEEE 7th Advanced Information Technology, Electronic and Automation Control Conference (IAEAC)*. Vol. 7. IEEE, 2024.
- [17] Lee, Minjae, et al. "A proposal on multi-agent static path planning strategy for minimizing radiation dose." *Nuclear Engineering and Technology* 56.1 (2024): 92-99.
- [18] Wang, Jiaqi, Ning Zhao, and Chao Mi. "ART path planning method based on the 3D steering angle weighted A-star algorithm." *International Jo*
- [19] Zhao, Xiaoru, et al. "Multi-UAV Path Planning and Following Based on Multi-Agent Reinforcement Learning." *Drones* 8.1 (2024): 18.
- [20] Cui, Junguo, et al. "Multi-strategy adaptable ant colony optimization algorithm and its application in robot path planning." *Knowledge-Based Systems* 288 (2024): 111459.
- [21] Janovská, Kristýna, and Pavel Surynek. "Multi-agent Path Finding in Continuous Environment." *arXiv preprint arXiv:2409.10680* (2024).
- [22] Ardizzoni, Stefano, et al. "An algorithm with improved complexity for pebble motion/multi-agent path finding on trees." *Journal of Artificial Intelligence Research* 79 (2024): 483-514.
- [23] Wu, Zixuan, et al. "Diffusion-Reinforcement Learning Hierarchical Motion Planning in Adversarial Multi-agent Games." *arXiv preprint arXiv:2403.10794* (2024).

- [24] Pohan, Muhammad Aria Rajasa, et al. "Path Planning using Combined Informed Rapidly-exploring Random Tree Star and Particle Swarm Optimization Algorithms." *IEEE Access* (2024).
- [25] Yang, Yibin, et al. "Attention-based Priority Learning for Limited Time Multi-Agent Path Finding." *Proceedings of the 23rd International Conference on Autonomous Agents and Multiagent Systems*. 2024.
- [26] Skrynnik, Alexey, et al. "Learn to Follow: Decentralized Lifelong Multi-Agent Pathfinding via Planning and Learning." *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 38. No. 16. 2024.
- [27] Wang, Wenjuan, et al. "Towards optimization of path planning: An RRT*-ACO algorithm." *IEEE Access* (2024).
- [28] Sim, Joonyeol, Joonkyung Kim, and Changjoo Nam. "Safe Interval RRT* for Scalable Multi-Robot Path Planning in Continuous Space." *arXiv preprint arXiv:2404.01752* (2024).
- [29] Fan, Honghui, et al. "BI-RRT*: An improved path planning algorithm for secure and trustworthy mobile robots systems." *Heliyon* 10.5 (2024).
- [30] Langari, Seied Mohammad, Faridaddin Vahdatikhaki, and Amin Hammad. "Improving the performance of RRT path planning of excavators by embedding heuristic rules." *Advanced Engineering Informatics* 62 (2024): 102724.
- [31] Zhao, Xiaoru, et al. "Multi-UAV Path Planning and Following Based on Multi-Agent Reinforcement Learning." *Drones* 8.1 (2024): 18.
- [32] Lee, Minjae, et al. "A proposal on multi-agent static path planning strategy for minimizing radiation dose." *Nuclear Engineering and Technology* 56.1 (2024): 92-99.
- [33] Alkazzi, Jean-Marc, and Keisuke Okumura. "A Comprehensive Review on Leveraging Machine Learning for Multi-Agent Path Finding." *IEEE Access* (2024).
- [34] Zhang, Lixiang, et al. "Multi-agent policy learning-based path planning for autonomous mobile robots." *Engineering Applications of Artificial Intelligence* 129 (2024): 107631.
- [35] Zhou, Yaozhe, Yujun Lu, and Liye Lv. "Grid-based non-uniform probabilistic roadmap-based agv path planning in narrow passages and complex environments." *Electronics* 13.1 (2024): 225.

- [36] Solis, Irving, et al. "Adaptive robot coordination: A subproblem-based approach for hybrid multi-robot motion planning." *IEEE Robotics and Automation Letters* (2024).
- [37] Kumar, Sunil, and Afzal Sikander. "A novel hybrid framework for single and multi-robot path planning in a complex industrial environment." *Journal of Intelligent Manufacturing* 35.2 (2024): 587-612.
- [38] Feng, Haolong, et al. "Method of Multi-agent Path Planning for Lunar Robots Swarm Based on Improved CBS Algorithm." *2024 43rd Chinese Control Conference (CCC)*. IEEE, 2024.
- [39] Moldagalieva, Akmaral, et al. "db-cbs: Discontinuity-bounded conflict-based search for multi-robot kinodynamic motion planning." *2024 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2024.
- [40] Tang, Mingkai, et al. "MGCBS: An Optimal and Efficient Algorithm for Solving Multi-Goal Multi-Agent Path Finding Problem." *arXiv preprint arXiv:2404.19518* (2024).
- [41] Zhou, Zhuo, et al. "A multi-AGV fast path planning method based on improved CBS algorithm in workshops." *Proceedings of the Institution of Mechanical Engineers, Part C: Journal of Mechanical Engineering Science* 238.5 (2024): 1507-1521.
- [42] Wu, Mingyu, et al. "Evaluation of Particle Swarm Optimization, Genetic Algorithms, and Ant Colony Optimization in Autonomous Mobile Robots Scheduling: A Comparative Study." *2024 IEEE International Conference on Industrial Technology (ICIT)*. IEEE, 2024.
- [43] Zhang, Zhen, et al. "Collaborate to Adapt: Source-Free Graph Domain Adaptation via Bi-directional Adaptation." *Proceedings of the ACM on Web Conference 2024*. 2024.
- [44] Khan, Junaid Ahmed, Weiyi Wang, and Kaan Ozbay. "BELIEVE: Privacy-Aware Secure Multi-Party Computation for Real-Time Connected and Autonomous Vehicles and Micro-Mobility Data Validation Using Blockchain—A Study on New York City Data." *Transportation research record* 2678.3 (2024): 410-421.
- [45] Mishra, Medhavi, Sumit Mishra, and Dongsoo Har. "Integrating Multi-sourced Sensor Data for Enhanced Traffic State Estimation." *IEEE Sensors Journal* (2024).

- [46] Cui, Can, et al. "A survey on multimodal large language models for autonomous driving." *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*. 2024.
- [47] Chen, Guojun, et al. "Path Planning for AUVs Based on Improved APF-AC Algorithm." *Computers, Materials & Continua* 78.3 (2024).
- [48] Kottinger, Justin, et al. "Introducing Delays in Multi Agent Path Finding." *Proceedings of the International Symposium on Combinatorial Search*. Vol. 17. 2024.
- [49] Zhang, Dengxing, Chen Chen, and Guanyu Zhang. "AGV path planning based on improved A-star algorithm." *2024 IEEE 7th Advanced Information Technology, Electronic and Automation Control Conference (IAEAC)*. Vol. 7. IEEE, 2024.
- [50] Sattar, Shoaib, et al. "Cardiac Arrhythmia Classification Using Advanced Deep Learning Techniques on Digitized ECG Datasets." *Sensors* 24.8 (2024): 2484.
- [51] Alparslan, Onder, and Omer Cetin. "Real-Time indoor path planning using object detection for autonomous flying robots." *Intell. Autom. Soft Comput* 36.3 (2023): 3355-3370.
- [52] Tang, Yunchao, et al. "Obstacle avoidance motion in mobile robotics." *Journal of System Simulation* 36.1 (2024): 1-26.
- [53] Zhang, Xiaotong, et al. "D-PBS: Dueling Priority-Based Search for Multiple Nonholonomic Robots Motion Planning in Congested Environments." *IEEE Robotics and Automation Letters* (2024).
- [54] Choi, Jaewoong, et al. "Environment-detection-and-mapping algorithm for autonomous driving in rural or off-road environment." *IEEE Transactions on Intelligent Transportation Systems* 13.2 (2012): 974-982.
- [55] Li, Jiaoyang, et al. "Lifelong multi-agent path finding in large-scale warehouses." *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 35. No. 13. 2021.
- [56] Tang, Junjie, et al. "Modeling Analysis of Carrier-based Aircraft's Scheduling on Flight Deck." *Journal of Physics: Conference Series*. Vol. 1746. No. 1. IOP Publishing, 2021.

- [57] Li, Jiaoyang, et al. "MAPF-LNS2: Fast repairing for multi-agent path finding via large neighborhood search." *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 36. No. 9. 2022.
- [58] Geft, Tzvika. "Fine-Grained Complexity Analysis of Multi-Agent Path Finding on 2D Grids." *Proceedings of the International Symposium on Combinatorial Search*. Vol. 16. No. 1. 2023.
- [59] Tian, Lijing, et al. "An improved rapidly-exploring random trees algorithm combining parent point priority determination strategy and real-time optimization strategy for path planning." *Sensors* 21.20 (2021): 6907.
- [60] Sharon, Guni, et al. "Conflict-based search for optimal multi-agent pathfinding." *Artificial intelligence* 219 (2015): 40-66.
- [61] Stern, Roni, et al. "Multi-agent pathfinding: Definitions, variants, and benchmarks." *Proceedings of the International Symposium on Combinatorial Search*. Vol. 10. No. 1. 2019.
- [62] Gordon, Ofir, Yuval Filmus, and Oren Salzman. "Revisiting the complexity analysis of conflict-based search: New computational techniques and improved bounds." *Proceedings of the International Symposium on Combinatorial Search*. Vol. 12. No. 1. 2021.
- [63] Ren, Zhongqiang, et al. "Binary branching multi-objective conflict-based search for multi-agent path finding." *Proceedings of the International Conference on Automated Planning and Scheduling*. Vol. 33. No. 1. 2023.
- [64] Okumura, Keisuke. "Lacam: Search-based algorithm for quick multi-agent pathfinding." *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 37. No. 10. 2023.
- [65] Chaddad, Ahmad, et al. "Electroencephalography signal processing: A comprehensive review and analysis of methods and techniques." *Sensors* 23.14 (2023): 6434.
- [66] Kottinger, Justin, Shaull Almagor, and Morteza Lahijanian. "Conflict-based search for explainable multi-agent path finding." *Proceedings of the International Conference on Automated Planning and Scheduling*. Vol. 32. 2022.

- [67] Sartoretti, Guillaume, et al. "Primal: Pathfinding via reinforcement and imitation multi-agent learning." *IEEE Robotics and Automation Letters* 4.3 (2019): 2378-2385.
- [68] Damani, Mehul, et al. "PRIMAL \$ _2 \$: Pathfinding via reinforcement and imitation multi-agent learning-lifelong." *IEEE Robotics and Automation Letters* 6.2 (2021): 2666-2673.
- [69] Okumura, Keisuke, et al. "Priority inheritance with backtracking for iterative multi-agent path finding." *Artificial Intelligence* 310 (2022): 103752.
- [70] Yakovlev, K. S., et al. "Planning and learning in multi-agent path finding." *Doklady Mathematics*. Vol. 106. No. Suppl 1. Moscow: Pleiades Publishing, 2022.
- [71] Honglin, Zhang, et al. "Collaborative optimization of task scheduling and multi-agent path planning in automated warehouses." *Complex & Intelligent Systems* 9.5 (2023): 5937-5948.
- [72] Tiwari, Saurabh. "Smart warehouse: A bibliometric analysis and future research direction." *Sustainable Manufacturing and Service Economics* 2 (2023): 100014.
- [73] Xie, Lin, Hanyi Li, and Laurin Luttmann. "Formulating and solving integrated order batching and routing in multi-depot AGV-assisted mixed-shelves warehouses." *European Journal of Operational Research* 307.2 (2023): 713-730.
- [74] Quadrini, Walter, Elisa Negri, and Luca Fumagalli. "Open interfaces for connecting automated guided vehicles to a fleet management system." *Procedia Manufacturing* 42 (2020): 406-413.
- [75] Lv, Peng, et al. "Optimal multi-robot path planning for cyclic tasks using Petri nets." *Control Engineering Practice* 138 (2023): 105600.
- [76] Cheng, Pangcheng David Cen, et al. "Path planning in formation and collision avoidance for multi-agent systems." *Nonlinear Analysis: Hybrid Systems* 47 (2023): 101293.
- [77] Yu, Jiabin, et al. "A traversal multi-target path planning method for multi-unmanned surface vessels in space-varying ocean current." *Ocean Engineering* 278 (2023): 114423.
- [78] Li, Baiyu, and Hang Ma. "Double-deck multi-agent pickup and delivery: Multi-robot rearrangement in large-scale warehouses." *IEEE Robotics and Automation Letters* 8.6 (2023): 3701-3708.

- [79] Lin, Qiushi, and Hang Ma. "SACHA: Soft actor-critic with heuristic-based attention for partially observable multi-agent path finding." *IEEE Robotics and Automation Letters* (2023).
- [80] Chen, Yu, et al. "Multi-UAV autonomous path planning in reconnaissance missions considering incomplete information: A reinforcement learning method." *Drones* 7.1 (2022): 10.
- [81] Okubo, Takuma, and Masaki Takahashi. "Multi-Agent Action Graph Based Task Allocation and Path Planning Considering Changes in Environment." *IEEE Access* 11 (2023): 21160-21175.
- [82] Jin, Jianzhi, et al. "Conflict-based search with D* lite algorithm for robot path planning in unknown dynamic environments." *Computers and Electrical Engineering* 105 (2023): 108473.
- [83] Hu, Hongtao, et al. "Anti-conflict AGV path planning in automated container terminals based on multi-agent reinforcement learning." *International Journal of Production Research* 61.1 (2023): 65-80.
- [84] Ji, Xuebo, et al. "Decentralized, unlabeled multi-agent navigation in obstacle-rich environments using graph neural networks." *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2021.
- [85] Blumenkamp, Jan, et al. "A framework for real-world multi-robot systems running decentralized GNN-based policies." *2022 International Conference on Robotics and Automation (ICRA)*. IEEE, 2022.
- [86] Li, Qingbiao, et al. "Graph neural networks for decentralized multi-robot path planning." *2020 IEEE/RSJ international conference on intelligent robots and systems (IROS)*. IEEE, 2020.
- [87] Ma, Hang. "Graph-based multi-robot path finding and planning." *Current Robotics Reports* 3.3 (2022): 77-84.
- [88] Li, Qingbiao, et al. "Graph neural networks for decentralized multi-robot path planning." *2020 IEEE/RSJ international conference on intelligent robots and systems (IROS)*. IEEE, 2020.
- [89] Dai, Hanqi, et al. "Cooperative path planning of multi-agent based on graph neural network." *2022 34th Chinese Control and Decision Conference (CCDC)*. IEEE, 2022.

- [90] Li, Qingbiao, et al. "Message-aware graph attention networks for large-scale multi-robot path planning." *IEEE Robotics and Automation Letters* 6.3 (2021): 5533-5540.
- [91] Chandra, Rohan, et al. "Socialmapf: Optimal and efficient multi-agent path finding with strategic agents for social navigation." *IEEE Robotics and Automation Letters* 8.6 (2023): 3214-3221.
- [92] Yang, Fan, et al. "Far planner: Fast, attemptable route planner using dynamic visibility update." *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2022.
- [93] Li, Jiaoyang, et al. "New techniques for pairwise symmetry breaking in multi-agent path finding." *Proceedings of the International Conference on Automated Planning and Scheduling*. Vol. 30. 2020.
- [94] Wang, Hanfu, and Weidong Chen. "Multi-robot path planning with due times." *IEEE Robotics and Automation Letters* 7.2 (2022): 4829-4836.
- [95] Wen, Licheng, Yong Liu, and Hongliang Li. "CL-MAPF: Multi-agent path finding for car-like robots with kinematic and spatiotemporal constraints." *Robotics and Autonomous Systems* 150 (2022): 103997.
- [96] Rahman, Mustafizur, et al. "An adaptive agent-specific sub-optimal bounding approach for multi-agent path finding." *IEEE Access* 10 (2022): 22226-22237.
- [97] Andreychuk, Anton, et al. "Multi-agent pathfinding with continuous time." *Artificial Intelligence* 305 (2022): 103662.
- [98] Semiz, Fatih, and Faruk Polat. "Incremental multi-agent path finding." *Future Generation Computer Systems* 116 (2021): 220-233.
- [99] Greshler, Nir, et al. "Cooperative multi-agent path finding: Beyond path planning and collision avoidance." *2021 International Symposium on Multi-Robot and Multi-Agent Systems (MRS)*. IEEE, 2021.
- [100] Solis, Irving, et al. "Representation-optimal multi-robot motion planning using conflict-based search." *IEEE Robotics and Automation Letters* 6.3 (2021): 4608-4615.
- [101] Ren, Zhongqiang, Sivakumar Rathinam, and Howie Choset. "Multi-objective conflict-based search for multi-agent path finding." *2021 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2021.

- [102] Lee, Hannah, et al. "Parallel hierarchical composition conflict-based search for optimal multi-agent pathfinding." *IEEE Robotics and Automation Letters* 6.4 (2021): 7001-7008.
- [103] Ren, Zhongqiang, Sivakumar Rathinam, and Howie Choset. "Conflict-Based Steiner Search for Multi-Agent Combinatorial Path Finding." *Proceedings of Robotics: Science and Systems*. 2022.
- [104] Li, Jiaoyang, et al. "Disjoint splitting for multi-agent path finding with conflict-based search." *Proceedings of the international conference on automated planning and scheduling*. Vol. 29. 2019.
- [105] Yuan, Jianya, et al. "A novel GRU-RNN network model for dynamic path planning of mobile robot." *IEEE Access* 7 (2019): 15140-15151.
- [106] Li, Minghang, Xiaozhen Yan, and Qinghua Luo. "Dynamic path planning for multi-handling robots in warehousing system." *2020 Global Reliability and Prognostics and Health Management (PHM-Shanghai)*. IEEE, 2020.
- [107] . Greshler, Nir, et al. "Cooperative multi-agent path finding: Beyond path planning and collision avoidance." *2021 International symposium on multi-robot and multi-agent systems (MRS)*. IEEE, 2021.
- [108] Chen, Xin, Yanjie Li, and Lintao Liu. "A coordinated path planning algorithm for multi-robot in intelligent warehouse." *2019 IEEE International Conference on Robotics and Biomimetics (ROBIO)*. IEEE, 2019.
- [109] Arulkumaran, Kai, et al. "A brief survey of deep reinforcement learning." *arXiv preprint arXiv:1708.05866* (2017).
- [110] Sichkar, Valentin Nikolaevich. "Comparison analysis of knowledge-based systems for navigation of mobile robot and collision avoidance with obstacles in unknown environment." *Информатика, телекоммуникации и управление* 11.2 (2018): 64-73.
- [111] Sichkar, Valentyn N. "Reinforcement learning algorithms in global path planning for mobile robot." *2019 International Conference on Industrial Engineering, Applications and Manufacturing (ICIEAM)*. IEEE, 2019.
- [112] Chu, Zhenzhong, et al. "Path planning based on deep reinforcement learning for autonomous underwater vehicles under ocean current disturbance." *IEEE Transactions on Intelligent Vehicles* 8.1 (2022): 108-120.

- [113] Beomsoo, Han, Ankit A. Ravankar, and Takanori Emaru. "Mobile robot navigation based on deep reinforcement learning with 2d-lidar sensor using stochastic approach." *2021 IEEE International Conference on Intelligence and Safety for Robotics (ISR)*. IEEE, 2021.
- [114] Rhazzaf, Mohamed, and Tawfik Masrour. "Smart Autonomous Vehicles in High Dimensional Warehouses Using Deep Reinforcement Learning Approach." *Engineering Letters* 29.1 (2021).
- [115] Yang, Yang, Li Juntao, and Peng Lingling. "Multi-robot path planning based on a deep reinforcement learning DQN algorithm." *CAAI Transactions on Intelligence Technology* 5.3 (2020): 177-183.
- [116] Zong, Chaofei, Xiaotong Yao, and Xiaoli Fu. "Path planning of mobile robot based on improved ant colony algorithm." *2022 IEEE 10th Joint International Information Technology and Artificial Intelligence Conference (ITAIC)*. Vol. 10. IEEE, 2022.
- [117] Wang, Jiankun, et al. "Neural RRT*: Learning-based optimal path planning." *IEEE Transactions on Automation Science and Engineering* 17.4 (2020): 1748-1758.
- [118] Lee, HyeokSoo, and Jongpil Jeong. "Mobile robot path optimization technique based on reinforcement learning algorithm in warehouse environment." *Applied sciences* 11.3 (2021): 1209.
- [119] Wang, Qiuyang, et al. "K-means based path optimization for smart warehouse system." *2021 International Conference on Advanced Mechatronic Systems (ICAMechS)*. IEEE, 2021.
- [120] Fan, Yunfeng, Fang Deng, and Xiang Shi. "Multi-robot task allocation and path planning system design." *2020 39th Chinese Control Conference (CCC)*. IEEE, 2020.
- [121] Teja, Pamarthi Ravi, and AA Nippun Kumaar. "QR code based path planning for warehouse management robot." *2018 International Conference on Advances in Computing, Communications and Informatics (ICACCI)*. IEEE, 2018.
- [122] Yang, Yang, Li Juntao, and Peng Lingling. "Multi-robot path planning based on a deep reinforcement learning DQN algorithm." *CAAI Transactions on Intelligence Technology* 5.3 (2020): 177-183.
- [123] Chang, Lu, et al. "Hierarchical multi-robot navigation and formation in unknown environments via deep reinforcement learning and distributed

- optimization." *Robotics and Computer-Integrated Manufacturing* 83 (2023): 102570.
- [124] Zhu, C. Y. "Intelligent robot path planning and navigation based on reinforcement learning and adaptive control." *J. Logist. Inform. Serv. Sci* 10.3 (2023): 235-248.
 - [125] Ubaidillah, Achmad, and Hanifudin Sukri. "Application of Odometry and Dijkstra Algorithm as Navigation and Shortest Path Determination System of Warehouse Mobile Robot." *Journal of Robotics and Control (JRC)* 4.3 (2023): 413-423.
 - [126] Ubaidillah, Achmad, and Hanifudin Sukri. "Application of Odometry and Dijkstra Algorithm as Navigation and Shortest Path Determination System of Warehouse Mobile Robot." *Journal of Robotics and Control (JRC)* 4.3 (2023): 413-423.
 - [127] Alhassow, Mustafa Mohammed, Oguz Ata, and Dogu Cagdas Atilla. "Car-like robot path planning based on voronoi and Q-learning algorithms." *2021 International Conference on Engineering and Emerging Technologies (ICEET)*. IEEE, 2022.
 - [128] Alhassow, Mustafa Mohammed, Oguz Ata, and Dogu Cagdas Atilla. "Multi-agents path planning for a mobile robot in a dynamic warehouse environment." *International Conference on Computing, Intelligence and Data Analytics*. Cham: Springer International Publishing, 2023.
 - [129] Alhassow, Mustafa Mohammed, Oguz Ata, and Dogu Cagdas Atilla. "Obstacle Avoidance Capability for Multi Target Path Planning in Different Style of Search." (2024).
 - [130] Alhassow, Mustafa Mohammed, Oguz Ata, and Dogu Cagdas Atilla "Nonholonomic Path Planning for A Mobile Robot Based on Voronoi and Q- Learning Algorithm", *International Conference on Computing and Machine Intelligence*, 2021.
 - [131] Y. Liang, L. Wang, "Applying genetic algorithm and ant colony optimization algorithm into marine investigation path planning model," **Soft Computing**, vol. 24, pp. 8199–8210, 2020.
 - [132] G. Chen, T. Wu, Z. Zhou, "Research on ship meteorological route based on A-star algorithm," **Mathematical Problems in Engineering**, vol. 2021, pp. 1–8, 2021.
 - [133] B. Fu et al., "An improved A* algorithm for the industrial robot path planning with high success rate and short length," **Robotics and Autonomous Systems**, vol. 106, pp. 26–37, 2018.

- [134] H. Lu et al., “Deep fuzzy hashing network for efficient image retrieval,” **IEEE Transactions on Fuzzy Systems**, vol. 29, no. 1, pp. 166–176, 2020.
- [135] T. XiangRong, Y. Z., X. XinXin, “Improved A-star algorithm for robot path planning in a static environment,” **Journal of Physics: Conference Series**, IOP Publishing, 2021.
- [136] A. Mavrogiannis, R. Chandra, D. Manocha, “B-gap: Behavior-rich simulation and navigation for autonomous driving,” **IEEE Robotics and Automation Letters**, vol. 7, no. 2, pp. 4718–4725, 2022.
- [137] L. Wen, Y. Liu, H. Li, “CL-MAPF: Multi-agent path finding for car-like robots with kinematic and spatiotemporal constraints,” **Robotics and Autonomous Systems**, vol. 150, 103997, 2022.
- [138] M. M. Alhassow, O. Ata, D. C. Atilla, “Car-like robot path planning based on Voronoi and Q-learning algorithms,” **2021 International Conference on Engineering and Emerging Technologies (ICEET)**, IEEE, 2021.
- [139] X. Zhang et al., “D-PBS: Dueling Priority-Based Search for Multiple Nonholonomic Robots Motion Planning in Congested Environments,” **IEEE Robotics and Automation Letters**, vol. 9, no. 4, pp. 638–639, 2024.
- [140] R. D'Andrea, “Guest editorial: A revolution in the warehouse: A retrospective on Kiva systems and the grand challenges ahead,” **IEEE Transactions on Automation Science and Engineering**, vol. 9, no. 4, pp. 638–639, 2012.
- [141] D. Zhang, C. Chen, G. Zhang, “AGV path planning based on improved A-star algorithm,” **2024 IEEE 7th Advanced Information Technology, Electronic and Automation Control Conference (IAEAC)**, IEEE, 2024.
- [142] K. Zhang, Z. Yang, T. Başar, “Multi-agent reinforcement learning: A selective overview of theories and algorithms,” **Handbook of Reinforcement Learning and Control**, pp. 321–384, 2021.
- [143] D.-H. Cho, D.-S. Jang, H.-L. Choi, “Memetic algorithm-based path generation for multiple Dubins vehicles performing remote tasks,” **International Journal of Systems Science**, vol. 51, no. 4, pp. 608–630, 2020.
- [144] A. Bakdi et al., “Optimal path planning and execution for mobile robots using genetic algorithm and adaptive fuzzy-logic control,” **Robotics and Autonomous Systems**, vol. 89, pp. 95–109, 2017.

- [145] R. Almadhoun et al., “A survey on multi-robot coverage path planning for model reconstruction and mapping,” **SN Applied Sciences**, vol. 1, pp. 1–24, 2019.
- [146] J. J. Roldán et al., “Heterogeneous multi-robot system for mapping environmental variables of greenhouses,” **Sensors**, vol. 16, no. 7, p. 1018, 2016.
- [147] H. Hu et al., “Anti-conflict AGV path planning in automated container terminals based on multi-agent reinforcement learning,” **International Journal of Production Research**, vol. 61, no. 1, pp. 65–80, 2023.
- [148] Y. Jeon, D. Park, “Poster: Adaptive A* Algorithm for Calculation Time Reduction of Autonomous Vehicle's Pathfinding,” **2024 IEEE Vehicular Networking Conference (VNC)**, IEEE, 2024.
- [149] J. Li et al., “MAPF-LNS2: Fast repairing for multi-agent path finding via large neighborhood search,” **Proceedings of the AAAI Conference on Artificial Intelligence**, 2022.
- [150] W. Hönl et al., “Persistent and robust execution of MAPF schedules in warehouses,” **IEEE Robotics and Automation Letters**, vol. 4, no. 2, pp. 1125–1131, 2019.
- [151] T. Guo, J. Yu, “Decentralized Lifelong Path Planning for Multiple Ackerman Car-Like Robots,” *arXiv preprint arXiv:2402.11767*, 2024.
- [152] I. Solis et al., “Representation-optimal multi-robot motion planning using conflict-based search,” **IEEE Robotics and Automation Letters**, vol. 6, no. 3, pp. 4608–4615, 2021.
- [153] A. Zanardi et al., “Factorization of Multi-Agent Sampling-Based Motion Planning,” **2023 62nd IEEE Conference on Decision and Control (CDC)**, IEEE, 2023.
- [154] Q. Lin, H. Ma, “SACHA: Soft Actor-Critic with Heuristic-Based Attention for Partially Observable Multi-Agent Path Finding,” **IEEE Robotics and Automation Letters**, vol. 8, no. 1, pp. 1–8, 2023.
- [155] L. Chang et al., “Reinforcement-based mobile robot path planning with improved dynamic window approach in an unknown environment,” **Autonomous Robots**, vol. 45, pp. 51–76, 2021.
- [156] E. S. Low, P. Ong, C. Y. Low, “A modified Q-learning path planning approach using distortion concept and optimization in a dynamic environment for an autonomous mobile robot,” **Computers & Industrial Engineering**, vol. 181, p. 109338, 2023.

- [157] F. H. Ajeil et al., "Autonomous navigation and obstacle avoidance of an omnidirectional mobile robot using swarm optimization and sensors deployment," **International Journal of Advanced Robotic Systems**, vol. 17, no. 3, p. 1729881420929498, 2020.
- [158] G. Fragapane et al., "Planning and control of autonomous mobile robots for intralogistics: Literature review and research agenda," **European Journal of Operational Research**, vol. 294, no. 2, pp. 405–426, 2021.
- [159] J. Yuan et al., "A novel GRU-RNN network model for dynamic path planning of mobile robot," **IEEE Access**, vol. 7, pp. 15140–15151, 2019.
- [160] H. Tang, "Multi-robot material delivery in industrial parks based improved on A* algorithm," **Highlights in Science, Engineering, and Technology**, vol. 46, pp. 280–288, 2023.
- [161] J. Li, W. Ruml, S. Koenig, "Eecbs: A bounded-suboptimal search for multi-agent pathfinding," **Proceedings of the AAAI Conference on Artificial Intelligence**, 2021.
- [162] R. Cui, J. Guo, B. Gao, "Game theory-based negotiation for multiple robots task allocation," **Robotica**, vol. 31, no. 6, pp. 923–934, 2013.
- [163] D. Zhang et al., "Deep reinforcement learning for visual object tracking in videos," arXiv preprint arXiv:1701.08936, 2017.
- [164] Q. Wu et al., "A new fallback beetle antennae search algorithm for path planning of mobile robots with collision-free capability," **Soft Computing**, vol. 24, pp. 2369–2380, 2020.
- [165] R. Chandra, D. Manocha, "Gameplan: Game-theoretic multi-agent planning with human drivers at intersections, roundabouts, and merging," **IEEE Robotics and Automation Letters**, vol. 7, no. 2, pp. 2676–2683, 2022.
- [166] Q. Li et al., "Graph neural networks for decentralized multi-robot path planning," **2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)**, IEEE, 2020.
- [167] Alhassow, Mustafa Mohammed, Oguz Ata, and Dogu Cagdas Atilla. "Obstacle Avoidance Capability for Multi-Target Path Planning in Different Styles of Search." *Computers, Materials & Continua* 81.1 (2024).