Republic of Iraq
Ministry of Higher Education and Scientific Research
University of Babylon

ROBOT PATH PLANNING BASED ON HYBRID ADAPTIVE DIMENSIONALITY REPRESENTATION WITH GLOWWORM SWARM OPTIMIZATION

A Dissertation
Submitted to the Council of the College of Information Technology for Postgraduate Studies of University of Babylon in Partial Fulfillment of the Requirements for the Degree of Doctor of Philosophy in Information Technology-Software

QASIM RADAM MAHMOOD ATHAB ALOBAIDI

Supervised by
Prof. Hussein Kietan Mansi Alwaan
Asst. Pro. Ali Hadi Hassan Abbas

2022A.D. **1443 A.H.**

Republic of Iraq
Ministry of Higher Education and Scientific Research
University of Babylon

بِسْمِ اللَّهِ الرَّحْمَٰنِ الرَّحِيمِ

وَ قُلْ رَبِّ زِدْنِي عِلْماً

صدق الله العلي العظيم

سورة طه **الآية** 114

2022A.D. **1443 A.H.**

Supervisor Certification

Supervisor Certification

I certify that this dissertation entitled " ***Robot Path Planning Based on Hybrid Adaptive Dimensionality Representation with Glowworm Swarm optimization Algorithms***" was prepared under my supervision at the department of Software / College of Information Technology / University of Babylon, by **Qasim Radam Mahmood** as a partial fulfillment of the requirements of the degree of **Doctor of Philosophy in Information Technology / Software**.

Signature:

Prof. Dr. ***Hussein K. Khafaji***

Assistant Prof. *Dr.* ***Ali Hadi Hasan*****.**

Date: /   /2022

The Head of the Department Certification

In the view of available recommendations, I forward the dissertation entitled " ***Robot Path Planning Based on Hybrid Adaptive Dimensionality Representation with Glowworm Swarm optimization Algorithms*** " for debate by the examination committee.

Signature:

Assistant Prof. Dr. ***Ahmed Saleem***

Title: **Head of Software Department, College of Information Technology, University of Babylon.**

Date**:   /   / 2022**

Certification of the Examination Committee

Signature:
Name: Dr. Abass H. Hassin Alasadi
Title: Professor
Date: / / 2022
(**Chairman**)

Signature:
Name: Dr. Khalid Ali Hussein
Title: Professor
Date: / / 2022
(**Member**)

Signature:
Name: Dr. Ahmed Saleem Abass
Title: Assistant Professor
Date: / / 2022
(**Member**)

Signature:
Name: Dr. Mehdi Ebadi Manaa
Title: Assistant Professor
Date: / / 2022
(**Member**)

Signature:
Name: Dr. Saif Ali Abd AlRadha Alsaidi
Title: Assistant Professor
Date: / / 2022
(**Member**)

Signature:
Name: Dr. Hussein K. Khafaji
Title: Professor
Date: / / 2022
(**Member and Supervisor**)

Signature:
Name: Dr. Ali Hadi Hasan
Title: Assistant Professor
Date: / / 2022
(Member and Supervisor)

Approved by the Dean of the College of Information Technology, University of Babylon.

Signature:
Name: Dr. Hussien Attiah Al-Khalidi
Title: Professor
Date: / / 2022
(**Dean of Collage of Information Technology**)

Acknowledgement

First and foremost, I would like to thank creator God, Almighty God, for making me aware of and for giving me graces endless. My gratitude to the beacon of science, master of creation, to the maximum of the prophet Mohammad (peace and Blessings of Allah Be Upon Him) who sent by Allah for all creatures.

I would like to show my greatest appreciation to my supervisors; Ass. pro. Dr. Ali Hadi Hassan and Pro. Dr. Hussin K. Khafaji. Words are inadequate and I can't say thank you enough for their tremendous support and help. Without their encouragement and guidance this research would not have materialized. Also, without the support of the head of software department Dr. Ahmed Salem Al-Safar this research would not have been possible.

Also, I would like to thank all my colleagues at the Babylon university who help me much to achieve this work. Mustafa Abd-Alrasol, Wahid Kadhum, Haidar Monthir, Zaid Kuba and so much others.

I wish to express my special thanks to my work colleagues at the Diyala university who help me much to achieve this work. Dr. Salam Abd-Alkhalik, Dr. Adeel Abd-Alwahab, Dr. Mohammad Sami, Dr. Faiza Abdul-Qadir, Dr. Sameera Abd-Alkadeer, Dr. Ehssan Salman, Ali Hussin Fathil, Dr. Atiaf Esmail, Basim Najm Al-Deen, Dr. Entisar Abid, Hussam Shaker, Rasha Sobhe and Assel Dawod.

Abstract

Now a day’s, robotics occupies considerable attention in human life especially in the modern industries that are automatically running. Robot path planning is considered a key problem for safe and efficient robot movement. On the other side, optimizing the robot velocity to move the mobile robot from an initial position to a goal position is a critical factor in the robotics filed. Robot velocity controlling is important in the following states. (1) When the robot navigates in the tight free area. (2) When the robot corner around the angles of its path heading to its goal. The adaptive dimensionality (AD) concept has been successfully used to solve this type of problem. On the other hand, algorithms of swarm intelligence optimization are largely successful in finding solutions to the robot's path planning problems.

The first aim of this research is to develop an approach for planning the robot's path in its dynamic state space based on hybrid adaptive dimensionality representation with Glowworm swarm optimization algorithms. While, the second aim of this research is achieved by developing the approach to adaptive the velocity of the robot based on the value of each rotation angle at each node of its path that resulted from the application of the first approach and the number of the neighboring obstacles that are surrounding this node within its closest two levels. The third aim of this research is determining an appropriate data structure to perform all the operations in the two proposed research approaches to reduce the required storage space and time occupation for processing.

The effectiveness of these two approaches are gaged by applying them on different dynamic environments to find the optimal robot path and then adaptive its velocity and then comparing these obtaining results with the results that gathered by applying hybrid mixed D∗ algorithm with Lbest PSO, the hybrid improved D* algorithm with Lbest PSO approaches. These comparisons prove the effectiveness of the proposed first approach in reducing the average number of iterations required to become (19.33) iterations, while for the same experiments the average of execution time for the two others algorithms are 84.16, 29.46 iterations respectively. Additionally, the average of path cost of the proposed algorithm is 19.27, which is the best value compared with the averages that are obtained by applying the two others algorithms which have been 84.17 and 29.47, respectively. Finally, the comparison results illustrate that the average of execution time is 4.81 seconds for the experiments of the proposed approach. While, for the same experiments the average of execution time for the two others algorithms are 23.10 second and 10.616 second respectively. It also significantly reduced the execution time. On the other hand, the robot's velocity rate is improved by a percentage of up to 300%, and this indicates the effectiveness of the second proposed approach.

Declaration Associated with this Thesis

I hereby declare that this dissertation, submitted as a dissertation to the Council of the College of Information Technology for Postgraduate Studies of the University of Babylon in Partial Fulfillment of the Requirements for the Degree of Doctor of Philosophy in Information Technology-Software has not been submitted as an exercise for a similar degree at any other university. I also certify that the work described here is entirely my own except for exercise and summaries whose sources are appropriately cited in the references.

This thesis may be available within the university library and may be photocopied or loaned to other libraries for consultation.

Qasim Radam Mahmood

February, 2022.

Table of Contents

List of Tables

List of Figures

List of Abbreviations

Abbreviation	Description
ACO	Ant Colony Optimization
AD	Adaptive Dimensionality
APF	Artificial Potential Field
APSO	Adaptive PSO
GA	Genetic Algorithm
GSO	Glowworm Swarm Optimization
IGSO	Improved Glowworm Swarm Optimization
IS	Intelligent Swarm
MVR	Maximum Visual Range
POMDP	Partially Observable Markov Decision Process
PSO	Particle Swarm Optimization
RRT	Rapidly-exploring Random Tree
UQ	Uncertainty Quantification

CHAPTER ONE

GENERAL INTRODUCTION

1.1 Introduction

As a modernistic system, robots need the capability to plan their actions in wide and complex environments with different degrees of freedom. The independent mobile robot has been located in a large number of applications comprising the industry, rescuing, agriculture, space, mining and military. Intelligence capabilities of these mobile robots can be considered as the key factor for their successful navigations. one of the most important intelligent features among these abilities is the path planning process for these robots. Based on these facts, the path planning problem is one of the most important and researched topics[1].

The robot path planning purpose is to find a mobile robot path that can be safe. On the other hand, this path is required to be optimal. Therefore, proposing a navigation approach characterized by efficiency for robots is one of the critical issues in autonomous robotics [2]. In other words, planning of path or the path finding is increasingly required after the wide proliferating of the mobile robots in the different fields so that, these potentials, significantly in the field of autonomous control have become an area of focus [3].

A geometric path is generated by a path planning algorithm to the final point, from an initial point, passing through viapoints that are predefined, either in the space that is joined to the space that the robot operates in it or in the robot operating space. Therefore, insuring safe and effective navigation from point to point, an appropriate algorithm for path planning needs to be chosen, and the algorithm that is used to obtain the optimal path relies on the geometry of the robot in additional to the constraints computing [4].

Navigation of high-speed is preferred for robots to increase the efficiency of the provided services. Several limitations should be taken into account in order to achieve the high-speed navigation of the robot, which can be summarized as follows:

1) limitations of dynamic and mechanical;
2) limitations of control and computational;
3) environment unexpected dynamic change.

Wheel slippage or rollover in the case of a sharp cornering or an emergency stop made by the robot is indicated by the first limitation. The speed of robot navigation is limited by speed of processing, cost of computations, and control response. In contrast, the assumption that there may be dynamic obstacles in the occluded areas is indicated by the third limitation. Therefore, this research will develop two approaches, first is to plan the robot path from the start state down to its ending destination[5].

1.2 Path planning

The Robot path planning goal is to determine the safe path of the robot within a given environment, which is optimal or semi-optimal. Describing the problem of the robot path planning can be summarized in [6][7]:determined a robot and its work space, searching to find the optimal or suboptimal path by the mobile robots from the start point to the goal state based on the specific criterion of performance. Since the mobile robot path planning has important application value. Also, it has become a hot research topic both at abroad and home. Algorithms of path planning are used by mobile robots, vehicles, unmanned aerial and autonomous cars to determine the safe, efficient, collision-free, and least-cost travel paths from an start point to a goal point [8]. On the other words, one of the most crucial research problems is the path planning in the robotic. Therefore, by proposing path planning, many problems in different fields are solved. To select suitable sequence of action, a simple path planning has been applied to guide the robot to reach a specific goal. The path planning process cannot be applied in advance in all cases since if the robot environment is global, its information is not always available a priori[6].

Also, what should be noted, a suitable path is created as a sequence of action to reach the goal state starting from the start state through many intermediate states. In the algorithms of path planning, every decision is selected based on the available information that were stored in the current state after that by using criteria such as the shortest distance measures to the goal state using Euclidean distance computation. From the start point to the goal point there may be many paths in the search space. On the other side, the target can’t be reached since there is no possible path. In many other cases, in optimization terms, the optimal path must be the shortest distance that lies in the free space and reach the goal point during shortest period of time[7].

1.3 Mobile robot navigation problems

Three types of a robot can be seen in the study of robot navigation; system of single robot, system of multi-robots, and swarm robots. In general, Multi-robot and swarm robots are different through the form and task of the specific system. A small number of a robot is used to represent the multi-robot. Each one of them has differents shapes and functions from the others. All these robots work together to achieve the same goal[9].

While, a substantial amount of simple robots that are similar in shape and tasks use to represent the swarm robots system, which works together to achieve their functions by using a local communication and coordination . In this state, in all mobile robots, a system of navigation is built to allow them for exploiting its abilities in sensing, processing, and actuating abilities to decide control. Anyways, in order to identify the case of robots that are moving together in an unknown surrounding environment to find the goal, the obstacles are created to be static and dynamic in both multi-robot and swarm robots systems. This also indicates that the robot can be a dynamic obstacle[9].

Therefore, if the implementing any of the previous robot systems occurs, it must have the ability to move in its real-time path with all the difficulties arising from the surrounding environment in order to reach the specified goal. However, many considerations connected to many issues related to implementation, uncertainties, imprecision, and incomplete information in unorganized surrounding of the real world[9].

one of the critical tasks in the navigation is the perception and cognition used to collect knowledge about the robot environment , and how the control commands are executed that are achieved through many sensors and actuators. Depending on the sensors process's perception and actuators' control process, a mobile robot navigation can be divided into four types: navigation based on map, navigation based on behavior, navigation based on learning, and navigation based on communication. Figure (1.1) illustrates all the navigational tasks, behaviors, and the robot types used in the navigation system [9].

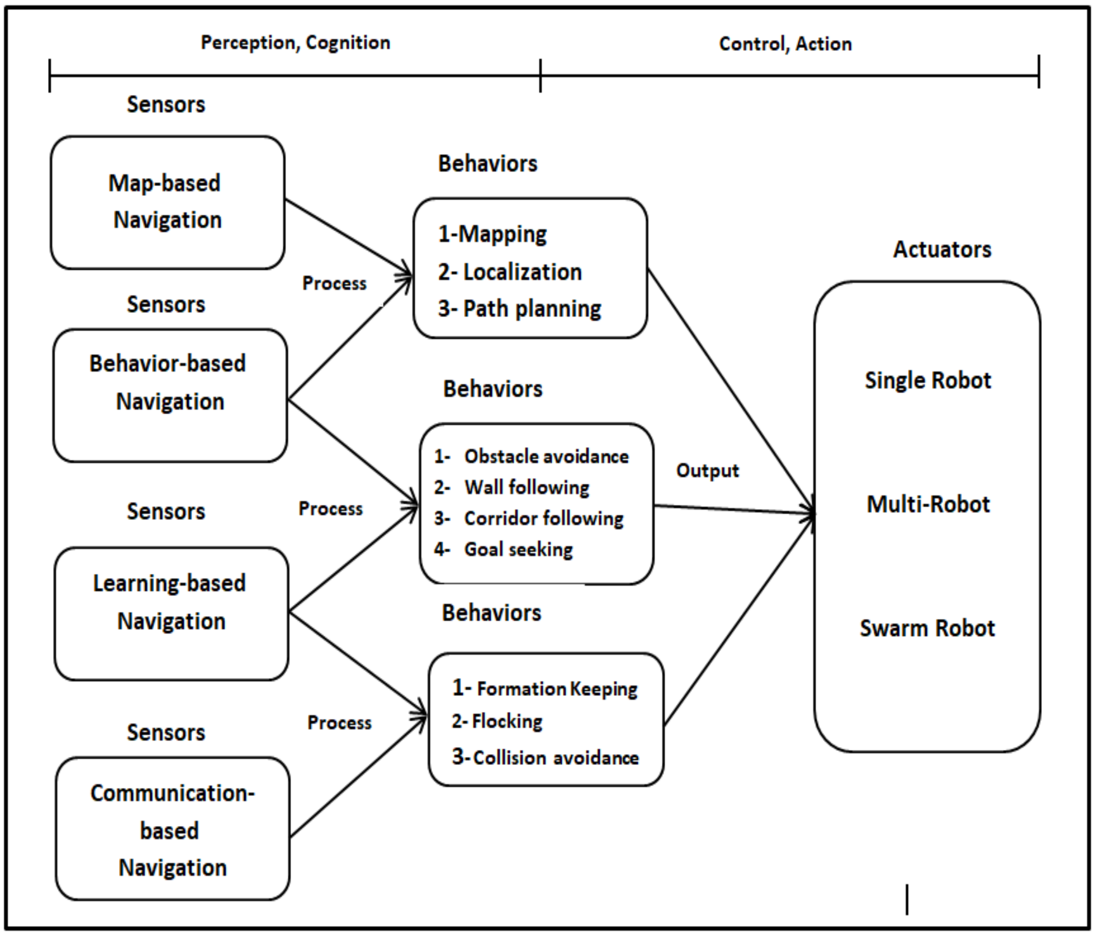


Figure (1.1): Navigation system based on perception-actuation

1.4 Related work

To give recommendations in various applications there are several publication designed such as the following researches:

1. Amalia et al. in [10] presents a methodology to avoid obstacle. This work is considered as an extension for another previous method in which the obstacle motion was be predicted as well as how this

prediction can be used during plan the robot trajectory down to the goal. This extension is rendered by deciding if a robot must change its velocity for avoiding an obstacle effectively. There are three different velocities: slow, normal and fast, the robot can select one of them to move. Controlling the robot movement is achieved by a Hierarchical Partially Observable Markov Decision Process (POMDP).

2. Yoshiro et al. in [11], used an adaptive robot velocity control approach for robot velocity control during its movement in its unknown environment. In this speed control method, the robot velocity controlling is needed in two states, when the robot moves in narrow free area and When a robot enters an area the free spaces that lies within it has not been decided yet. The approach that is proposed in this paper is simple but an effective for velocity control when the robot selects the fastest safe velocity. The proposed approach improved the navigation efficiency of robot three times with the same safety.
3. L. He, and S. Huang in [12], the improved glowworm swarm optimization (IGSO) is presented to improve the quality of finding the global optimal value efficiently and accurately according to the standard GSO algorithm. In this work a new algorithm is proposed to update the step size. Moreover, it extends the sensor range to the whole search space. Also, it introduces the random movement of the brightest glowworms of the firefly algorithm. Later, the IGSO will use different objective functions for the multilevel color image thresholding problem.
4. K. Gochev in [13], the framework of planning with AD is used to make an effective use of the state abstraction and the reduction of

dimensionality in order to reduce the size and complexity of the state-space. A hybrid state-space consisting of both the abstracted and the non-abstracted states is frequently constructed and searched by a hybrid state-space. At first, only the abstracted states are in the state space and then the regions of the non-abstracted states are selectively introduced into the state-space to keep the resulting path feasibility and the algorithm, theoretically, guarantees the completeness and bounds on solution cost sub-optimality.

5. A. Hadi in [14], based on a Max-Min Ant Colony optimization(ACO) algorithm improved version proposed an approach to identify an optimal local path in the robot's dynamic environment. In this proposed method, to build class nodes that include the space environment information, the grid method in order to decompose two-dimensional space is required to be used.

   In Max-Min ACO algorithm, the desired development occurs in the mixing pheromone trail updating phase with the strategies of D* algorithm in order to create the consequence modified (deposited) pheromone trail update at each iteration. Therefore, the robot environment will be analyzed starting from the goal point(food)until the start point(nest)to determine the cost (pheromone deposition) for all the nodes. Then, the tour construction probabilities that are obtained will be used by the mobile robot to select the best solution in order to navigate starting with specific point called start state to reach the ending point (goal) through its dynamic environment and avoiding all the types of obstacles[14].

6. A. T. Sadiq in [15], the hybrid D* algorithm with Lbest PSO was used to find the optimal path of the robot between initial and goal states. This approach offers two improvements. In the first improvement, the D* algorithm is used to compute the cost for each node by analyzing the robot environment starting with goal node reaching to the start point of the robot path. While the Lbest PSO algorithm role is moving the robot starting from specific point (start state) until the goal through its environment, path re-planned process must be achieved when the gate raise state appears because the robot unable to pass this node during it's trying to determine the optimal path.

   The second improvement of this approach offers a solution method for processing the gate raise state by discovering it in the analysis stage, closing this node, making it an obstacle node, and make it as a high cost value state. This process is unlike the re-planning step required with the D* algorithm.

7. Girija [16] produced a new hybrid algorithm using particle swarm optimization(PSO) with Artificial potential field(APF) in order to fast path planning in the robot environment that has huge number of obstacles. The proposed approach is designed for improving the velocity of finding robot path with path cost that is feasible and minimal cost in an obstacle-rich environments by combining an PSO and APF. The proposed hybrid approach creates the path points to be followed based on a PSO and APF combination and this path is found In robot environments of different density in terms of the number of obstacles. The efficient of a new hybridization is gaged by comparing it with the basic PSO. The results prove that the new hybrid approach has the ability to reach the lowest possible cost of the route in much less time compare with the original PSO. On the other side, the obtained results illustrated hybrid approach even the robot move in the dynamic environment.

1.5 Problem statement

- The movement of mobile robots from a specific point to a desired target is a challenge to complete the tasks assigned in this dissertation. The challenges and the problems in this dissertation are identified as follows:

- The first challenge is how to find the path of the robot from the starting point to the target point within its dynamic environment. Robot design needs to trade cost and speed in order to get reasonable price and speed regardless of the type of robot. On the other hand, one of the most important obstacles in the robot path planning process is the identification of factors and dimensions and their adaptation within each area of the robot's environment to ensure a safe and rapid movement to the robot from the starting state to the target state.

- The second issue is how to adaptive the robot velocity to make it move at the highest possible safe velocity; within its environment that surrounds its path taking into consideration all the factors that can limit its velocity specifically, angle's value at each path's node and the number of obstacles that are surrounding this node towards the robot motion

- The last challenge is how to determine an appropriate data structure to perform all the operations in the previous two points. This helps to reduce the required storage space, and the occupation time needed to accomplish these tasks.

1.6 Research contributions

This research has three main contributions, the following points illustrate them.

1. This study develops a hybrid method for path planning based on Adaptive Dimensionality with glowworm optimization algorithm.
2. The research study develop an efficient method to adapt the velocity of a robot on its obtained path based on the number of obstacles surrounding each path node in the closest two levels and the angle value at it
3. This study, also develop a method of data structure to the state space and the values of dimensions for each represented unit in the space.

1.7 Structure of Dissertation

The first chapter, give the introduction, problem statement, motivation, contribution, related work, and structure of the dissertation. The rest of this dissertation is:

- **Chapter Two:** This chapter provides the introduction of robot path planning, a thorough overview of robot environment modeling with Robot path planning. Also, this chapter presents Path Planning

Categories. On the other hand, this chapter illustrates different types of Robot path planning algorithms. Also, this chapter addressed the concept of adaptive dimensionality in one of its parts and presents number of algorithms that based on this concept to plan the mobile robot path. Finally, the last part of this chapter focus on Mobile robot velocity controlling and adaptation it to suit the robot environment nature.

- **Chapter Three:** focuses on the proposed approaches design in details and the methods that are used.

- **Chapter Four:** represents the simulation and results of the proposed approaches which are discussed in details.

- **Chapter Five:** discusses the conclusions and the suggested future works.

Chapter Two
THEORETICAL FOUNDATION

2.1 Introduction

There are many types of automated environments in which mobile robots can work. Fully autonomous robots have attracted a lot of attention in recent decades and have developed remarkably, especially with regard to industrial robots and service robots in public places. The problem of robot path planning within its environment can be described as the mobile robot searches to find the optimal or suboptimal path from the start point into goal point relying on a specific performance criterion.

On the other hand, good technologies of path planning for the robot are used to save more of time but as well as can be used to reduce the wear and capital investment of the robot. Mobile robot path planning has important value of application therefore; it has become a hot research topic [15]. Controlling the velocity of the robot movement during its motion from a specific point to the final destination to perform its tasks optimizes it and makes its movement smoother in proportion to the nature of its environment [16].

2.2 Robot Environment Modeling

Models of environment are the key matter to independent decisions for mobile robot, and as well as the cooperation among set of robot that operates in the same environment. Such decisions that an autonomous robot or a team of robot can make relate to movements, perceptions, and communications. Thus, different kinds of environment models should be found [17].

In the Framework Space Approach, the robot is usually represented as a dot to simplify the problem, and the obstacles in its neighborhood are scaled. These two steps allow the mobile robot more freedom to move in the space of obstacles without collision both the obstacles and its environment boundary. This type of environment modeling approach involves a visibility graph, Voronoi graph and tangent graph [15].

While, the Free Space Approach is used to create free space within the robot environment and between the obstacles in terms of the free convex area based on the principle of free link. This graph is called MAKLINK which is created to produce generation of a conflict-free path. Using a set of midpoints of common free links among convex region as the passing points. The nodes are represented by these points, while the arcs are represented by the connection among the points in every convex region[18][20].

However, The main idea of other method which is called Topological Method to reduce the dimensions and transfer the problem of path planning to the discriminant problem of connectivity in low dimension from the high dimensional geometry space. After establishing the topology network, the mobile robot planning path is obtained from the initial state the target. This approach only needs less time for the model building and also less storage space comparing with the cell decomposition approach. While the topological method focused on the number of obstacles[19].

The last method to environment modeling is called approach decomposes. In this environment modeling the mobile robot workspace into regions, after that every one of these regions is generally named as a cell. a connected graph is formed from these grids and a path is searched from the start grid to the goal grid. Then, By the cells' ordinal number, the path is represented. Also, The two types of the cell decomposition approach are represented by the approximate cell decomposition and exact cell decomposition. In the exact cell decomposition, the space without an obstacle is decomposed into n of non-overlapping units. Then the space that generated by connecting these n units should be similar to the original free space. While, in the type of approximate cell decomposition, all of the grids are in a predetermined shape such as rectangular shape. Into a number of larger rectangles, the whole of robot environment is decomposed, every one of these rectangles is continuous. If there is any obstacle or boundary in the big rectangle, this rectangle should be divided into 4 small rectangular, with all the larger grid, this operation should be executed and there is need to repeat it until it arrives the boundaries of solution. This type of environment model is used to present the work space of robot in the tow proposed approaches of this dissertation[15][21].

2.3 Robot path planning

Path planning has attracted attention since the 1970s, path planning has attracted attention, and in the following years it has begun to be used in solving problems across various fields, from simple spatial path planning to determining the suitable sequence of actions that is needed to reach specific target. In both of entirely known environments or partially known in addition to 22 fully unknown environment in which the information collects from system-mounted sensors and update environmental maps inform the robot by the desired movements[22].[23].

One of the most important topic of research in robot mobile field is robot path planning. It can be summarized by how to determine the optimal path or suboptimal path for mobile robot from initial point in to goal without collision while satisfying certain optimization criteria. Finding the robot path of collision free through obstacles cluttered in a workspace is vital to design an autonomous path planning of robot.. On the other side, Before the mobile robot tries to move in its environment to accomplish its tasks, one of the key problems that is required to be resolved is path planning [6].

The path planning problem can be described in several steps which as follows: determine a robot type and its attributes with the environment that works on it, the robot searches to determine an optimal or suboptimal path from the specific point to the goal depending on a specific performance criterion[24].

2.3.1 Path Planning Categories

In this section, a classification of the different problems associated to mobile robot’s path planning is given. there are three classes based on the knowledge of robot’s that it has about the environment, the nature of environment, and the approach used to solve the problem as depicted in Figure (2.5) [25].

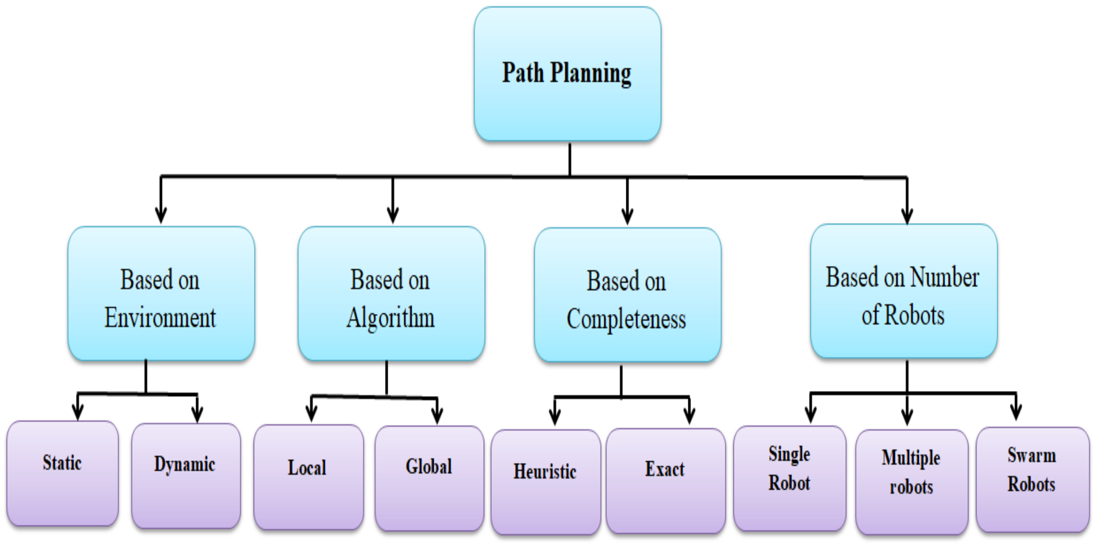


Figure (2.5) Path planning categories [25]

A. Environment Nature: In both static and dynamic environments, the problem of path planning is need to be done. The information of static environment is still fixed , where the locations of both start state and goal state are fixed, and over the time the obstacles don't change their positions. during due to the environment uncertainty[26].

Therefore, the path planning algorithm should have ability to be adapted with the suddenly and unexpected changes in the robot environment such as appears a new moving obstacle or when the goal is changed its position continuously. in this case, robot path planning process becomes more critical and there is need to effectively react in real time to movements of both goal and obstacle[27].

The environment that the robot works in it in the two improvement approaches of this dissertation will be as dynamic one.

B. Map knowledge: The process of robot path planning mainly depends on existing environment map which is as a reference to determine the positions of both start, goal and the points between them that use to form the robot path. The path planning algorithm designing process depend on the amount of knowledge that is represented on the map which plays the critical role in this process. Thus, path planning can be splited to two categories The robot has an priori information about the map of environment in the first class and this class of path planning is called as global path planning or deliberative path planning. While, the path planning in the second category depend on the fact that the mobile robot does not have priori knowledge about the environment that will work in it (i.e., uncertain environment). Accordingly, there is need to create an environment's estimated map based on sensing the locations of the obstacles in real time during the process of search for avoiding obstacles and acquiring an appropriate path toward the destination point. This path planning type is called local path planning or reactive navigation[25].

C. Completeness: The algorithms of path planning can be categorized into two classes; exact or heuristic depending on its completeness. An exact algorithm search to determine an optimal robot path if there is one exist or prove no one is exist. While, the heuristic one searches to find solution with good quality in short period of time. on the other hand, based on the number of robots that work in the same environment to achieve the same task, the path planning can be classified into three classes; single-robot, multi-robot, and swarm robot[25]. Glowworm swarm optimization algorithm that is used to move the robot on its environment work is one of the met-heuristic algorithms.

D. Number of Robots: In several robot applications, in the same environment, multiple robots work together. Thus, this type of problem is called multi-robot path planning. There are three types of robots that can be seen in their working environments. A small number of a robot is used to represent the multi-robot. Each one of them has differents shapes and functions from the others. All these robots work together to achieve the same goal. While, in swarm robots a huge amount of simple robots that are similar in shape and tasks are used to represent this system, which works together to achieve their functions by using a local communication and coordination . In this state, in all mobile robots, a system of navigation is built to allow them for exploiting its abilities in sensing, processing, and actuating abilities to decide control. Anyways, in order to identify the case of robots that are moving together in an unknown surrounding environment to find the goal, the obstacles are created to be static and dynamic in both multi-robot and swarm robots systems. This also indicates that the robot can be a dynamic obstacle. Also, there is single robot system in which only one robot moves from start state into goal state to find the optimal or suboptimal path[28].

2.4 Robot path planning algorithms

A robotic system Navigation includes different techniques and tools to gather information. In particular, one of the planning algorithms which are likely to be used to assists in initializing the robot location and identifying the next move. Navigation in the robot world incorporates detection and avoidance the object. In general, algorithms of path search to plan the path can be classified in to three classes: heuristic approaches and artificial intelligence algorithms and hybrid approaches. The following parts will illustrate that. Figure (2.6) shows these path planning techniques categories[29].

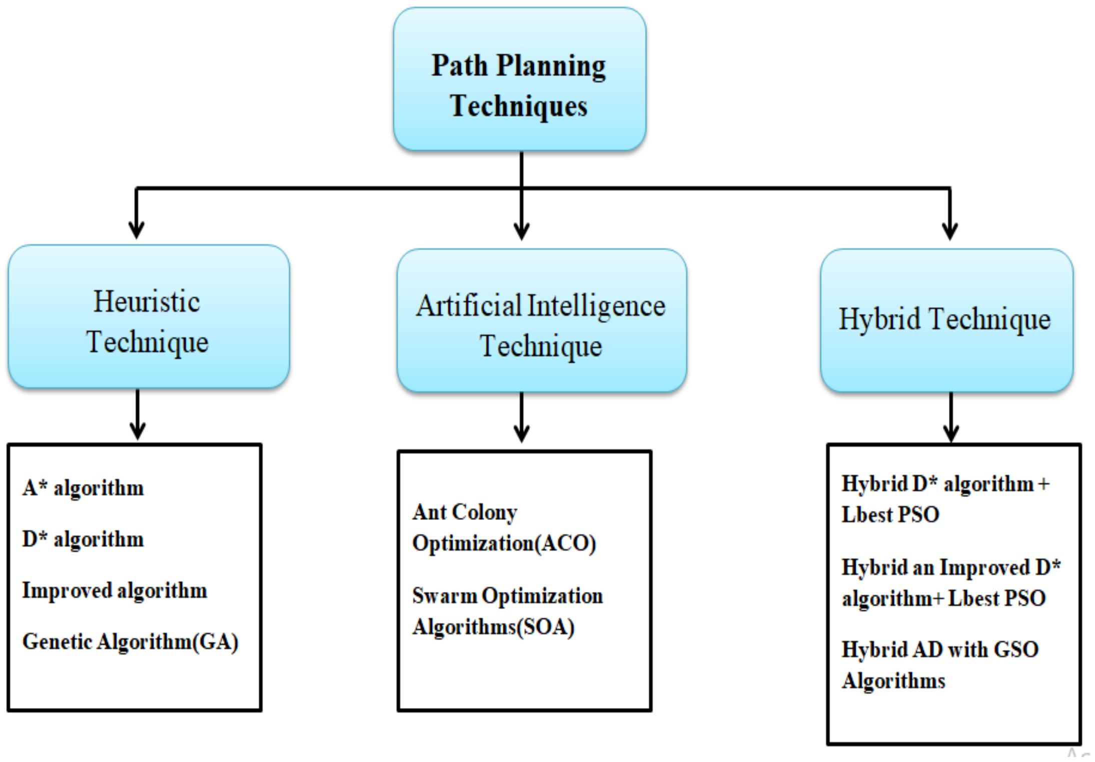

Path Planning Techniques
Heuristic Technique
Artificial Intelligence Technique
Hybrid Technique
A* algorithm
D* algorithm
Improved algorithm
Genetic Algorithm(GA)
Ant Colony Optimization(ACO)
Swarm Optimization Algorithms(SOA)
Hybrid D* algorithm + Lbest PSO
Hybrid an Improved D* algorithm+ Lbest PSO
Hybrid AD with GSO Algorithms

Figure 2.6 Path planning techniques categories

2.4.1 Heuristic Technique

Depending on the heuriskein which is a Greek verb that means (to find) The heuristic was derived which is considered faster than mathematical optimization (branch & bound, simplex, etc.). solving problems in a faster and more efficient method compared with classical ways by sacrificing optimality, accuracy or completeness for the sake of speed is consider as one of the key reasons that strongly pushed to design the heuristic algorithm. It is often times used to solve decision problems’ class called NP-complete problems. However, a solution can be verified when given but there is no known effective method to find a solution quickly in this type of problem.

The need for heuristic algorithms arises when exact solutions are necessarily computationally expensive and there is a possibility to suffice with the approximate solutions. Many heuristic algorithms such as Dijkstra, A* algorithm, and D* algorithm. Only D* algorithm is presented from this path planning category[30].

A- D* Algorithm

The D* algorithm was presented to generate an optimal traverse within the real-time through a graph with hanging or updated arc costs. Based on the fact that the most corrections of arc cost occur nearest to the robot, so that D* algorithm D* took this advantage and It only needs to be redesigned the path portion that lies between the corrections and robot’s current location. D* sets conditions for determining when a repair can be stopped, either because a new path was found or because the old path is still optimal.

One of the common algorithms that can be used to solve problems of mobile robot motion planning is D* algorithm. On the other hand, D* algorithm can be effectively use to solve problems that are related with optimization of path cost where the parameters of path cost during the traverse of the solution. There is number of path planning algorithms which are improved versions of the D* algorithm such as Focused D*[8], and D* Lite[31].

B- A* algorithm

A star(A*) is considered as one of the search method that use a function of heuristic (h(n)), where n is a node n. For each node n is associated an approximation h(h) of path cost from node until the target point, while h ∗ (n) represents the real distance (cost) starting from n node until the target state. A heuristic h is fixed if and only if: (if n is the goal node) where h(n) = 0, and (2) for all nodes and their successors n(0), the estimated cost of moving to the goal node from node n is not greater than the cost of moving to node n(0) from node n plus the estimated cost of moving to the goal node from node n(0) as can be shown in Equation (2.1).

$$h(n) <= c(n, n') + h(n') \ldots\ldots\ldots.(2.1)\ [15]$$

A heuristic h is acceptable when h(n) underestimates h ∗ (n), that is, it respects Equation (2.2), in order to be used the heuristic may be the straight line distance, or the Euclidean distance.

$$h(n) <= h * (n)\ldots\ldots\ldots\ldots\ldots\ldots\ldots(2.2)[15]$$

Also, in this method, there are another functions to achieve its sub tasks. Function f(n) to determine the cost of the path from the start node to node n is g(n). While, f(n) denotes the estimated cost of the path passing through n to reach the goal node. The sum of g(n) with h(n) is used to define the f(n) as in Equation (2.3)[32]

$$f(n) = g(n) + h(n)\ \ldots\ldots\ldots\ldots(2.3)[32].$$

2.4.2 Artificial Intelligence Technique

One of the most important contents for artificial intelligence and robotics is the problem of mobile robot path planning. Its task is to make the robot move independently from the starting state to the target state within its working environment, taking into account some movement restrictions. In the following parts, some artificial intelligence approaches which are used for robot path planning will be illustrated [19].

B. Swarm Optimization Algorithms

Swarm intelligence(SI) can be considered as a subset of artificial intelligence(AI). It was coined for first time in 1989 by Gerardo Beni and Jing Wang for the development of cellular robotic systems. The increasing popularity of swarm intelligence algorithms is due to several reasons. One of the main reasons is the versatility and flexibility that this type of algorithms provides to solve variety of problems[33].

Also, one of the main features that distinguished this type of algorithms is its ability to adapt with the external differences, and its ability to self-learning gave it the possibility of application in different areas of application. It is worth noting, that one of the problems that swarm intelligence algorithms contributed to effectively finding solutions to it is the mobile robot path planning problems, as will be discussed later. In the following parts number of swarm intelligent algorithms will be illustrated[34].

1. Ant Colony Optimization

Based on the foraging behavior of real ants for finding the source of food and transporting it to its nest, an optimization algorithm which called Ant Colony Optimization (ACO) was inspired. On implementing a homogeneous approach in which all ants depict similar 'behavioral traits' , most of research in ACO was focused. After the real ants succeed in reaching the food source and then return to their nest, during which they lay pheromones on the ground[35].

Based on this chemical pheromone that was placed on the ground, other ants are induced to reach the target place, and this mechanism is considered an indirect means of communication. Since the intensity of the pheromone is inversely proportional to the length of the path that the ant travels between the nest and the food source that it found therefore, the concentration of pheromone for the short path will be higher compared to the longer path. Based on this method, ant colony can identify a short path from the nest to the food source, and they can transport resources effectively. The transition probability in the ant colony algorithm can be calculated by using the following equation[36]:

$$p_{i,j}^{k} = \frac{(\tau_{i,j})^{\alpha}(\eta_{i,j})^{\beta}}{\sum_{l \in N_i^k}(\tau_{i,j})^{\alpha}(\eta_{i,j})^{\beta}}, \ if \ j \in N_i^k \quad \ldots\ldots\ldots\ldots\ldots(2.4)$$

Where $p_{i,j}^{d}$ represent the transition probability in which ant k will traverse from node i to node j, $(\tau_{i,j})^{\alpha}$ represents the intensity of the pheromone trail between nodes i and j with a corresponding weight value ofα, $(\eta_{i,j})^{\beta}$ represents the heuristic information between nodes i and j with corresponding weight value of β . $\eta_{i,j}$ $=1/d_{i,j}$ where $d_{i,j}$ represents the distance between nodes i and j. While Pheromone evaporation is implemented by the following equation:

$$\tau_{i,j} = (1-\rho)\tau_{i,j} \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots(2.5)$$

Where 0 < ρ ≤1 represents the pheromone evaporation rate. To avoid unlimited accumulation of the pheromone trails and it enables the algorithm to "forget" bad decision previously taken, the parameter ρ is used . The pheromone value can be updated by using the following equation:

$$\tau_{i,j} = \tau_{i,j} + \sum_{k=1}^{m} \Delta\tau_{i,j}^{k}, \forall (i,j) \in L \ldots\ldots\ldots\ldots\ldots\ldots (2.6)$$

Where $\Delta\tau_{i,j}^{k}$ represents the pheromone amount of ant k deposits on the paths it has visited? It is defined as follows:

$$\Delta\tau_{i,j}^{k} = \begin{cases} 1/C^{k}, \text{if path } (i,j) \text{ belongs to } T^{k}; \\ 0, \text{ otherwise;} \end{cases} \ldots\ldots\ldots\ldots\ldots\ldots(2.7)$$

Where C^{k} , the length of the tour T^{k} build by the k^{th} ant is computed as the sum of the lengths of the paths belonging to T^{k}.

2. Particle Swarm Optimization algorithm(PSO)

PSO is one of nature inspired algorithms, since its behavior is based on the social behavior of the flock of birds. It is a technique of an evolutionary computation designed by Kennedy and Eberhart. Firstly, it is started with the population that includes random solution. Secondly, the algorithm searches to find the optimum solution via updating generations. Thirdly, depending on the old generations reproduction will be achieved. Through the search space, a swarm of individuals called particles is flying in a PSO system[37].

A population of particles is maintained by the algorithm, where a potential solution to an optimization problem are represented by this population of particles. Also, by the best position visited by itself (i.e. its own experience) and the best particle position in the its neighborhood (the neighboring particles experience), the current particle position is influenced[38].

Also, when the neighborhood of a particle is the entire swarm, the best position in the neighborhood is referred to as the global best particle, and the resulting algorithm is referred to as a gbest PSO. When smaller neighborhoods are used, the algorithm is generally referred to as a lbest PSO. The basic PSO algorithm is illustrated in Algorithm (2.1).
It should be noted that the PSO algorithm was used in effective ways to find the solutions to the path planning problems of the mobile robot. In the next part, three of these algorithms will be illustrated[39].

X. Li *et. al* in [40], produced an Adaptive PSO (APSO) algorithm to find a solution for the problem of path planning of robot. This new algorithm is smarter compared with the conventional PSO. The objective function in this algorithm take into account the distance between the robot to the goal and obstacle respectively. While, Dewanga and *et. al* [41], produced an improved PSO (IPSO) with better optimal results which led to less the number of iteration steps comparing with other algorithms. IPSO algorithm reduced the iteration steps to find the optimal path about 20 iteration steps which led to a reduction in both of the length of the path and simulation time.

```
Algorithm PSO()
Initialize the algorithm parameters(c1, c2, W, Swarm_Size, Max_Iter).
for all i ∈ [1, Swarm_Size] do
    randomize $x_i(1)$, $v_i(1)$
    let $y_i(1) = x_i(1)$
    let $y'_i(1) = x_i(1)$
end for
for all t ∈ [1,Max_Iter] do
    compute $y_i(t)$ using equation (3.1)
    if (f ($y_i(t)$< min( f ( $y'_i(t-1)$, f ( f ($y_i(t)$)))
    then gbest=I and ( $y'_i(t) = y_i(t)$
end for
if the termination criterion( i.e Max_Iter)is met, then stop
for all i ∈ [1,Swarm_Size] do
    for j ∈ [1,N] do
        compute $v_i(t)$ using equation (3.3)
        if ($v_{i,j}(t+1)>v_{max}$) then clamp it to $v_{i,j}(t+1) = v_{max}$
        compute $x_{i,j}(t+1)$ using equation (3.4)
    end for
    end for
end
```

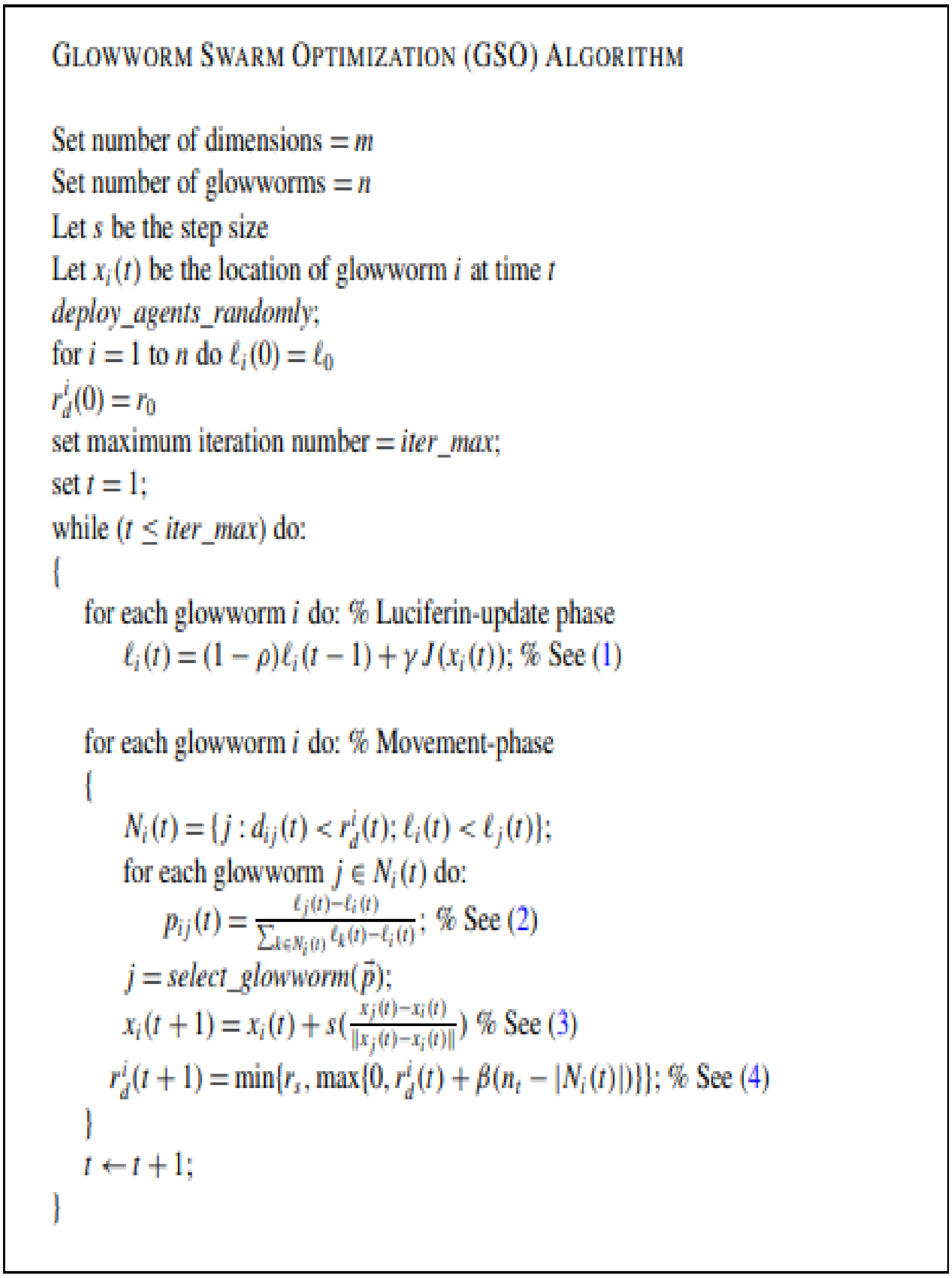
Algorithm (2.1) Basic PSO algorithm[39]

After conducting many different simulation experiments, which demonstrated the ability of the APSO algorithm to avoid obstacles and reach the goal in a short time, compared to the conventional PSO algorithm. M. Shahab and et. [42], proposed an algorithm for path planning based on PSO algorithm to find the shortest path that is collision-free in static environment. The optimal path is found using the proposed algorithm by performing random sampling on grid lines generated between the robot start state and goal state.

3. Glowworm Swarm Optimization(GSO)

The Glowworm Swarm Optimization (GSO)is one of the modern and original swarm intelligence algorithm which is designed for optimization by Krishnanand and Ghose in 2005 which simulate the glowworms flashing behavior in GSO algorithm, each glowworm is deemed as a feasible solution of target problem in space. Also, every glowworm has amount of luminescence named luciferin, which is identified using the glowworm’s current location function value. Along the movement path, the glowworm selects its neighbors depend on area of local-decision then it moves to the neighbor that has higher luciferin value than its own based on a probabilistic mechanism [43] [44].

Originally, algorithm of GSO was considered as algorithm for problems of optimization that applied directly to the tasks of collective robotic. The GSO algorithm has many characteristics in common with both the Ant Colony Optimization (ACO) algorithm and the Particle Swarm Optimization (PSO) algorithm but with some clear differences [45].



```
GLOWWORM SWARM OPTIMIZATION (GSO) ALGORITHM

Set number of dimensions = $m$
Set number of glowworms = $n$
Let $s$ be the step size
Let $x_i(t)$ be the location of glowworm $i$ at time $t$
deploy_agents_randomly;
for $i = 1$ to $n$ do $\ell_i(0) = \ell_0$
$r_d^i(0) = r_0$
set maximum iteration number = iter_max;
set $t = 1$;
while ($t \leq iter\_max$) do:
{
    for each glowworm $i$ do: % Luciferin-update phase
        $\ell_i(t) = (1 - \rho)\ell_i(t - 1) + \gamma J(x_i(t))$; % See (1)

    for each glowworm $i$ do: % Movement-phase
    {
        $N_i(t) = \{j : d_{ij}(t) < r_d^i(t); \ell_i(t) < \ell_j(t)\}$;
        for each glowworm $j \in N_i(t)$ do:
            $p_{ij}(t) = \frac{\ell_j(t) - \ell_i(t)}{\sum_{k \in N_i(t)} \ell_k(t) - \ell_i(t)}$; % See (2)
        $j = select\_glowworm(\vec{p})$;
        $x_i(t + 1) = x_i(t) + s\left(\frac{x_j(t) - x_i(t)}{\|x_j(t) - x_i(t)\|}\right)$ % See (3)
      $r_d^i(t + 1) = \min\{r_s, \max\{0, r_d^i(t) + \beta(n_t - |N_i(t)|)\}\}$; % See (4)
    }
    $t \leftarrow t + 1$;
}
```

Algorithm (2.2) Basic GSO algorithm[46]

- **The Standard GSO Algorithm**

GSO algorithm consists of four stages [44]:

i. Phase of glowworms distribute: In this phase, way based on randomly distributed may be used to distribute a collection of n glowworms in a search space at different locations. Now, an equal quantity luciferin value is given for all glowworms.

ii. Luciferin-update phase: Both of the value of an objective function and the previous luciferin level are used for the luciferin updating process. The luciferin update rule is:

$$\boldsymbol{l_i(t+1)=(1-p)l_i(t)+\gamma \textbf{Fitness}(x_i(t+1))} \;\ldots\ldots\ldots\ldots\; (2.8)$$

Where l (t)$_i$ point to the value of luciferin for glowworm i at t represent the number of iteration; γ and ρ are the enhancement constant and the luciferin decay, Fitness(x_i(t+l)) denotes to the objective function value at the glowworm i's location.

iii. Phase of glowworms movement: in the midst of the movement phase, by using a probabilistic mechanism every glowworm should be decided to move toward one of its neighbors that is the owner of luminosity than its own neighbors. For each glowworm, i the probability of moving toward a neighbor j is given by

$$\boldsymbol{P_{ij}(t) = \frac{l_j(t)-l_i(t)}{\sum_{k\in N_i(t)} l_k(t)-l_i(t)}} \;\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\; (2.9)$$

$$\boldsymbol{N_i(t) = \{j: (t) < r_i^d(t)\,;\, l_i(t) < l_j(t)\}} \ldots\ldots\ldots\ldots\ldots(2.10)$$

where, r$_{ij}$ (t) point to the Euclidean distance between the glowworms i and j at time t, and r$_i$ (t)d represents the variable neighborhood range related with glowworm i at $\boldsymbol{t_{th}}$ iteration. Then, the glowworm movements model can be stated as:

$$\boldsymbol{x_i}\,(\text{t} + 1) = \boldsymbol{x_i}\,(\text{t}) + \boldsymbol{s} \times \frac{(\boldsymbol{x_j}\,(\boldsymbol{t}) - \boldsymbol{x_i}\,(\boldsymbol{t}))}{\| \boldsymbol{x_j}\,(\boldsymbol{t}) - \boldsymbol{x_i}(\boldsymbol{t}) \|} \quad \ldots\ldots\ldots\ldots\ldots\ldots(2.11)$$

iv. Neighborhood range update phase: every glowworm starts with an initial neighborhood domain. After that at each GSO algorithm iterations the glowworm's neighborhood domain is updated by using specific rule.

$$r_i^d\ (t + 1) = \min\{r_s\,, \max\,\{0, r_i^d(t) + \beta(n_t - |N_i\,(t)|)\}\}\ldots\ldots\ldots(2.12)$$

where, β is a constant, n_t is a parameter to control the neighborhoods' number , r_s point to the sensory radius of the glowworm, $N_i\ (t)$point to the set of neighborhoods. The basic GSO algorithm is illustrated in Algorithm (2.2)[46].

2.4.3 Hybrid Technique

To solve problems in various fields, recently, many meta-heuristic algorithms have been proposed to find the global optimum value of a given problem. Therefore, Various operators to achieve exploration and exploitation attributes these algorithms must have them. Thus, their performances based on the control parameters selection and adjustment. Success in solving many real-world optimization problems critically depends on choosing the appropriate optimization algorithm with well-control parameters [47–49].

Generally, a better performance may result from the algorithm to solve particular problem, while when applicating same algorithm to find the solution for other problems, it may produce must worse performance. To avoid these difficulties number of approaches that based on the several techniques' combination were proposed which are called hybrid approaches. These models are very effective in optimization [50–52].

In the other words, an algorithm that combines more than one algorithm that solve the same problem is called hybrid algorithm. Generally, this process is performed to combine the desired individual features of both in order to obtain an algorithm that is much better than the individual ones. The following parts illustrate the many approaches proposed in literature often depend on some techniques' combination[53].

1. The hybrid D* algorithm with Lbest PSO

The hybrid D* algorithm with Lbest PSO was used by Sadiq in [61] to find the optimal path of the robot from the start point to another point called the goal. This approach offers two improvements. In the first improvement, the D* algorithm is used to compute the cost for each node by analysing the environment from the target node reaching the initial point. While, to move the robot from the start point until the goal point through its environment, the Lbest PSO algorithm is being used[54].

Path re-planned process must be achieved when appearing the gate raises state because the robot is unable to pass this node while its trying to find the optimal path. while the second improvement of this approach offers a solution method for processing the gate raise state by discovering it in the analysis stage, closing this node, making it an obstacle node, and a high value for the cost is given to it; unlike the re-planning step required with the D* algorithm.

2. Hybrid an improved D* with lbest PSO algorithm

In the hybrid an improved D* algorithm with lbest PSO algorithm, an improvement has been achieved by increasing the original 8 directions connections in the D* algorithm to 16 directions for exploring the neighborhood grid cells. Thus, the robot smallest gain angle decreases to π/8. Consequently, the cost of the path will be reduced [55].

3. An Adaptive PSO for Unconstrained Optimization.

Algorithm of adaptive particle swarm optimization for optimization that unconstrained, which it has ability to find with an expected quality, the different locations and defend its exploration-exploitation based on location of individual. The adapting learning is implemented by this algorithm where, the particles are pushed to achieve a natural conditioning behavior on an unconditioned motive. In the other hand, in problem space for this algorithm, into many categories the particles are divided therefore each particle wants for moving ahead to its best personal experience if it lies within category With a low diversity. while, if the particle lies in the category with high diversity, it will tend to move toward the category's global optimum. The birds' sensitivity idea to flying space of them is also used for increasing their speed in undesired spaces to escape from them as soon as possible[56].

4. A multilevel threshold image segmentation algorithm based on GSO

In the Glowworm Swarm Optimization Algorithm for Multi-Threshold Image Segmentation, the performance of standard GSO was enhanced improve the thresholding accuracy and the resultant robustness. This algorithm achieved these improvements by making adjustments on the local radius of the decision, the method of selection, and the step size. Also, the proposed algorithm illustrates an improved results, specially in terms of the larger numbers of thresholds compared with their counterparts in the standard GSO algorithm[57]

5. Fast Hybrid PSO-APF Algorithm for Path Planning in Obstacle Rich Environment.

Girija [16] produced a new hybrid algorithm using particle swarm optimization(PSO) with Artificial potential field(APF) in order to fast path planning in the robot environment that has a huge number of obstacles. The proposed approach improves the velocity of finding robot path with a feasible and minimum cost in an obstacle-rich environment by combining an PSO and APF.

The proposed hybrid approach creates the path points to be followed based on a PSO and APF combination, and this path is found in a robot environment with different obstacle densities. The efficiency of a new hybridization is gaged by comparing it with the basic PSO. The results prove that the new hybrid approach has the ability to reach the lowest possible cost of the route is much less time compared with the original PSO. On the other side, the obtained results illustrated hybrid approach even if the robot moves in a dynamic environment[16].

2.5 Planning with Adaptive Dimensionality(AD)

The adaptive dimensionality principle has been used recently with robot path planning algorithms to accelerate the planning process to find the optimal path in the search space. Therefore, variety of the algorithm overcomes the global planning in the state space of high-dimensional. The planning providing process in every algorithm is divided into process of two layers an input to a high-dimensional local planner, while a global planner deals with a low-dimensional state-space which avoided barriers locally so that they are fast and effective[58].

Nevertheless, these approaches may lead to paths that are highly suboptimal or unexecuted because of contradiction in the assumptions made by the global and local planners. In this paper, approach of path planning provides a method that calculate the values of an accurate heuristic that can be used later for steering the planning with high-dimensional by solving problem with low-dimensional which guarantee a feasible solution with respect to model of the high dimensional motion with making very fast progress in areas that appearing only low-dimensional structure. The AD approach has been extended by introducing an incremental planning algorithm in order to get faster planning times [59–60].

2.5.1 Planning with AD Publicly Accessible Penn Dissertations.

The framework for Planning with Adaptive Dimensionality is used by Gochev [15] for making an effective use of the state abstraction and the reduction of dimensionality in order to reduce the size and complexity of the state-space. A hybrid state-space consisting of both the abstract and the non-abstract states is frequently constructed and searched by a hybrid state-space consisting of both the abstract and the non-abstract states.

At first, only the abstract states are in the state space and then the regions of the non-abstract states are selectively introduced into the state-space to keep resulting path feasibility and the algorithm strong theoretical guarantees completeness and bounds on solution cost sub-optimality[15].

2.5.2 AD algorithm-based PCE for models with many model parameters

Y. Li *and et al.* in [62] presented an algorithm of adaptive dimension which based on PCE technique. Then the feasibility of the proposed approach will verify via the use solid rocket motor ignition under influence low temperature. the key approaches of this work can be summarized as follows: presenting dimension-adaptive algorithm includes two steps; through the PCE parameters computing using the algorithm of dimension-adaptive, Then the obtained PCE alternate model its accuracy is improved, and the proposed method to uncertainty

quantification (UQ) be applied for solid rocket motor ignition under low temperature for gaging the proposed method feasibility.

2.5.3 Efficient path planning for high-DOF articulated robots with AD.

Kim *and et al*. [63] proposed a path planning approach for high-degree of freedom (DOF) articulated robots with adaptive high-degree of freedom (DOF)dimensionality. In order to plan the path in C-space (configuration space) with high-dimensionality, first an adaptive body selection needs to be described that depends on it to select the bodies of robot and joints based on the path planning complexity.

Depending on that, to achieve a task of path planning, the robot may use necessary DOF. The configuration space with adaptive dimensionality for path planners based on sampling is built by method of The adaptive body selection. Then, introducing the adaptive Rapidly-Exploring Random Tree (RRT) algorithm via adaptive body selection[63].

2.6 Kinematic of particles

Particle’s kinematics can be defined as the geometrical study of their motion. It is used to connect displacement, velocity, acceleration, and time without paying attention to the causative factor of motion. On the other hand, Kinetic is the process of studying the connection between body mass, the forces acting on a body, and the body motion. To predict the motion caused by specific forces or compute the forces needed to produce a certain motion, kinematics is also used [64].

2.6.1 Rectilinear Motion of particles (Position, velocity and acceleration).

When a particle moves along a straight line, it is moves in a rectilinear motion. For example, the particle will occupy a particular location at any instant (t) on the straight line. The location of the particle, which is called the particle position coordinate is defined as the distance (position) x, with suitable sign[64].

Now, the instantaneous velocity of any position x at time the following Equation(2.13) is used:

$$\mathrm{v} = \frac{dx}{dt} \dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots (2.13)[64]$$

While, to compute the value of the instantaneous acceleration, Equation (2.14) is used to obtained that.

$$\mathrm{a} = \frac{dv}{dt} \dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots..\dots\dots (2.14)[64]$$

For computing both particle velocity and acceleration in rectilinear motion with constant acceleration respectively, the following two equations are used, which are as follows:

$$\mathrm{v} = \frac{ds}{dt} \Rightarrow s = s_0 \; + \; \mathrm{v}_0\, t + \frac{1}{2}\; a_c\, t \dots\dots (2.15)[64]$$

$$\mathrm{a} = \frac{d\mathrm{v}}{dt} \Rightarrow \mathrm{v} = \; \mathrm{v}_0 \; + \; a_c\, t \dots\dots\dots\dots\dots\dots\dots\dots.. (2.16)[64]$$

Where: v is the particle velocity, a is the particle acceleration, t represent the time, s is the travelled distance.

2.6 Mobile robot velocity controlling and adaptation

Nowadays, one of an important element in human life is robotics. This is due to the robot to successfully perform many keys and dangerous human activities [65]. Controlling the velocity of the robot movement during its motion from a specific point to the final destination to performing a task makes its movement smoother in proportion to the nature of its environment [66].

Velocity control produces smoother motion for the robot, and it curbs collision forces more effectively [67]. The next part presents many of these approaches that deal with this situation.

2.7.1 Predictive Control of Robot Velocity to Avoid Obstacles in Dynamic Environments.

Amalia et al. in [68] presented a methodology to avoid the obstacle. This work is considered an extension of another previous method in which the obstacle motion was predicted how this prediction can be used during plan the robot trajectory down to the goal. This extension is rendered by deciding if a robot must change its velocity for avoiding an obstacle effectively. There are three different velocities: slow, normal and fast, the robot can select one of them to move. Controlling the robot movement is achieved by a Hierarchical Partially Observable Markov Decision Process (POMDP).

2.7.2 Adaptive Robot Speed Control by Considering Map and Localization Uncertainty.

An adaptive robot velocity control approach is used by Yoshiro *and et al.* [69], for robot velocity control during its movement in its unknown environment. In this speed control method, the robot velocity controlling is needed in two states, when the robot moves in a narrow free area when a robot enters an area the free spaces not been decided yet. The approach that is proposed in this paper is simple but effective for velocity control when the robot selects the fastest safe velocity. The proposed approach improved the navigation efficiency of the robot three times with the same safety.

2.7.3 The Advantages of Velocity Control for Reactive Robot Motion.

Using velocity control as a clear choice, Andy et al. [70] provided an approach to reactive robot trajectory modification and online planning of motion. Smoother movement can be obtained by controlling the speed of the robot, which is more convenient to low control frequencies, more sturdy for signal variation's control as well as it is more effectively to decrease the forces of collision. Inherent safety regarding signal delays is the main advantage of position control.

This approach offers mathematical, graphical and experimental evidence for the points. The results of experiments depicted the efficiency of velocity control regarding to the low-frequency control signals and delay of signal. A cooperative manipulation task performed with stiff industrial manipulators is what the experiment relied on.

2.7.4 Selecting Optimum Path with Safe Speed Control by mobile robot considering the Blind Area of Vision Sensors.

Tsubasa et al. [71] used a speed control for the safe moving of an autonomous mobile robot in an environment with blind areas. Without controlled speed, a robot moving in such blind areas will crush the obstacle when it suddenly appears in the path. In this method, depending on the distance between the mobile robot and the blind areas in addition to the obstacles the safety of robot's velocity is neatly controlled. Despite the increase in the execution time, an optimal path is created to reduce moving time costs. Confirmation of the effectiveness of the proposed method is achieved by conducting experimenting using a real robot and computer simulation.

2.7.5 A velocity control strategy for collision avoidance of autonomous agricultural vehicles

Jinlin *and et al.* [72] proposed a method for velocity controlling to avoid collision by adjusting the speed of independent agricultural vehicles. This velocity adjustment process is based on the state of movement, degree of danger for the obstacle and the distance between the vehicles and obstacles. The strategy of this control has two steps: prediction of collision in a dynamic environment as well as an improved grid map of obstacle space–time, and collision avoidance velocity generator with a cloud model. the simulation results illustrate efficient of the proposed approach for predicting collision with anti-disturbance ability for threatening-free obstacle with rapid and accurate velocity output.

Chapter Three

PROPOSED SYSTEM

3.1 Introduction

this chapter proposes two design structure improvement approaches are proposed to find the optimal path of swarm robotic path planning and robot velocity optimization, respectively, in a dynamic environment. In every proposed approach in this study, a structure block is designed to show the scheme of the overview and detailed information related to the basic stages of processing; and to display how these phases are connected.

All the key processing phases are designed in the first proposed improvement approach. In the second proposed improvement approach, the design focuses on the robot movement phase that differs from its counterpart in the first development approach.

3.2 The Improved Planning Algorithm

The first improved approach of this study includes five key phases. It is designed to find the optimal path using hybrid adaptive dimensionality representation with a GSO algorithm starting with the start state to reach the goal state. Adaptive dimension is utilized with environment analyses starting with the goal node to compute each node cost ending with the start node. Through the dynamic environment, the GSO algorithm is used to move the mobile robot from the start state to the goal state. At the same time, encompassing dynamic obstacles is moving in the robot environment's free space through the optimal path finding and displaying. All the processing phases of the first development approach of this research study are illustrated in Figure (3.1).

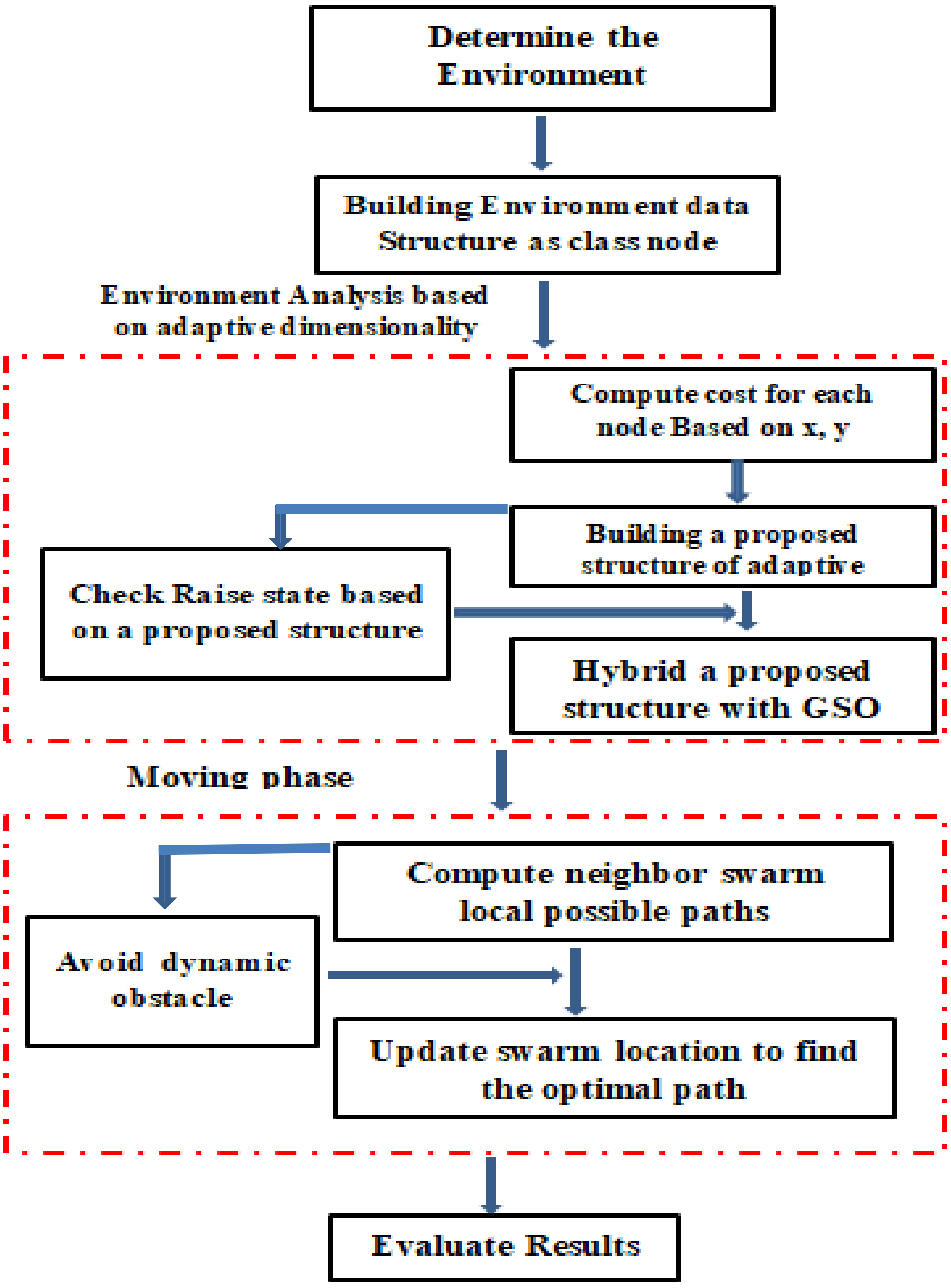

Determine the Environment
Building Environment data Structure as class node
Environment Analysis based on adaptive dimensionality
Compute cost for each node Based on x, y
Building a proposed structure of adaptive
Check Raise state based on a proposed structure
Hybrid a proposed structure with GSO
Moving phase
Compute neighbor swarm local possible paths
Avoid dynamic obstacle
Update swarm location to find the optimal path
Evaluate Results

Figure (3.1) First development approach's phases

First Phase: from the environment robot library store any image of environment in the key storage is selected by determining a path to find this environment, then as regular-grids cell environment, it’s displayed on a screen.

Second phase: In this phase each environment image cell is decomposed by converting it to a class node in array.

Third phase: This phase is known as environment analysis; it bases on adaptive dimension with several steps started with computing the cost of each node in this environment based on (x, y) dimensions. After that, building the proposed structure of an adaptive dimension and checking the raise state are involved depending on the implemented proposed structure.

Forth phase: Dynamic obstacles that are created randomly in free spaces of robot environment and are moved in any random neighbor positions in the free space. The role of the fourth phase is to avoid these dynamic obstacles.

Fifth phase: By using hybrid AD with GSO algorithms, the mobile robot moves from the start state in its dynamic environment avoiding both static and dynamic obstacles to identify the optimal path.

Final phase: In this phase, the results of the first approach of this dissertation are evaluated by comparing them with the results of other two literature approaches.

3.2.2 Converting the Image environment to Array of class nodes

The second phase of the methodology structure aims to convert the robot environment to an array of classes "nodes"; the nodes encompass all the information needed, and all the dynamic information obtained from the image environment is stored in a dynamic array of the nodes previously defined. By analyzing each cell of the image environment, the coordination position is defined as height and width, tag, and status. At the first click, the tag becomes new, while the status should have one of the following cases:

- Clear if clicking on free space.
- Obstacle if clicking on an obstacle.
- Start if clicking on start state.
- The goal if clicking on goal state.

The steps of this phase are achieved by calling Algorithm (3.1).

```
Algorithm (3.1): Convert image to node
Input: Bmp Image, state, Pixelcolor
Output: image environment as array of nodes
Begin
    1. for each grid cell in Hight
        1.1. for each grid cell in the Width {
                //make the state of all node as new
            1.1.1. state = newnode()
    2. for each grid cell in Hight
        2.1 for each grid cell in the Width
            2.1.1. state.tag =”New”
            2.1.2. state.ni = Hight.position
            2.1.3. state.nj = Width.position
            2.1.4. check  if the pixelcolor is black Then state.status=
                   “obstacle”
            2.1.5. if the pixelcolor is  red then state.status= “Start”
            2.1.6. if the pixelcolor is  gold then state.status= “Goal”
            2.1.7. else state.status= “Clear”}
End Algorithm
```

Algorithm (3.1) to convert image to array of class node

3.2.3 Environment analysis based on adaptive dimension

The third phase known as environment analysis based on adaptive dimension includes several steps. The first step is computing the cost of each node in this environment based on (x, y) dimensions. After that, it also involves building the proposed structure of an adaptive dimension and checking the raising state depending on the implemented structure. The last step of this stage includes the proposed hybrid structure with the GOS approach. The step of the analysis phase is needed to compute the cost consumed between the current node and its expanded neighbors.

When the neighbor status is an obstacle, its cost takes the highest value (10000). Also, there is a need to check if the state is in a horizontal or a vertical position of the current state when it lies in clear space (a space without obstacles). Thus, the current state cost increases by 1; otherwise, it increases by 1.4, Also, there is a need to check if the state is in a horizontal or a vertical position of the current state when it lies in clear space (a space without obstacles). Thus, the current state cost increases by 1; otherwise, it increases by 1.4, because the two horizontal , and vertical distances between any two nodes are equal to1, while the diagonal distance is 1.4 based on Euclidean distance rule. To spread information of the arc cost values when they are changed, the OPEN list is used. It is also used to calculate the cost of the path for the nodes within the search space. Through the repeated states removal process from the OPEN list the propagation takes place. A state is deleted from the available list every time it expands to pass the changes of cost to its neighbors. This process is continued by turning these neighbors places into the OPEN list. In this phase, three algorithms are illustrated.

```
Algorithm( 3. 2): Analysis compute cost
Input: image environment as array of nodes
Output: analysis cost value for each node for environment array
Begin
  1. computing the cost starts from goal state.
     1.1.  Goal. i = goal. position. X
     1.2.  Goal. j = goal. position. Y
     1.3.  Goa l. luc=0.0;  // luc=luciferin
     1.4.  Insert(OpenList, goal)
  2. Compute the cost for the other nodes
     2.1. node X=  DequeueOpenList ()
     2.2. X.tag = “Close”
     2.3. Find the eight neighbors of X       % call Algorithm(3.5)
     2.4. for each Y neighbors of X  {
          2.4.1. compute Cost( X, Y, OpenList)
          2.4.2. SortQueue(OpenList) }
     // continue until the stop condition is satisfied
     2.5 continue until reaching start state
End Algorithm
```

Algorithm(3. 2): to analysis the compute cost

The first step is achieved by putting a goal state as a current node in the queue OPEN list(Insert Algorithm). By finding the neighbors of the current node that lies at the closest level, the current node has been expanded Algorithm (3.2), and the second step of this phase is achieved.

The arc cost is calculated using the Algorithm (3.3) in the third step, where each node within the neighborhood gives a high cost value with a status of the state obstacle value 10000.

```
Algorithm (3.3): Compute node cost
Input: current node and one of its 8 neighbors
Output: the value of neighbor node cost(Ncost)

Begin
  1- Check if the neighbor node is new node
  2- Check the status of the neighbor node //clear, goal, start,
                                             obstacle.
     2.1- if the status is obstacle then neighbor Ncost=10000
     2.2- if the status is clear
          .2.2.1 if the neighbor lies in the same row or same
                 column of  current node
             Then neighbor Ncost= current node cost+1
             Else  neighbor Ncost= current node cost+1.4
  3. If the neighbor node status ≠start state go to step 1
End Algorithm
```

Algorithm (3.3) to compute the node cost

The neighbor needs to check its position compared to the location of the current state if it is clear. Then, it adds 1.4 to the arc cost if it lies on the diagonal of the current state; otherwise, it adds 1. After that, the current node becomes a back pointer for each state in the neighborhood.

Also, the luciferin value for each path node is calculated using 3-6 Algorithm (3.4). Then, the neighbors of the current state are transferred to the OPEN list while the current state is placed in the closed list. Sort Queue algorithm is saved in the queue OPEN list. This process continually expands until the start state is found.

```
Algorithm (3.4): Compute node luciferin value

Input: current node(X) and its 8 neighbors(N)
        enhancement constant ($\gamma$=0.6) and luciferin decay ($\rho$=0.4)
Output: the value of neighbor node luciferin(luc)

Begin
  1- Compute (luc) value by using the Ncost value that is
     calculated by  node cost Algorithm(3-1)
  2- N.luc= =(1- $\rho$) X. luc)+ ($\gamma$s.k)
  3. If the neighbor node status $\neq$start state go to step 1

End Algorithm
```

Algorithm (3.4) to compute node luciferin value

Gate Raise State algorithm is used to avoid gate rise state. When this case occurs, path replant is required because the robot size is greater than the clear space between vertices and static obstacles.

3.2.4 Moving Robot by Hybrid GSO with Adaptive Dimension

In this approach, the robot moves from the start state and finds the next specific solution using the GSO algorithm with an adaptive dimension. This is achieved by using Algorithm (3.5) which encompass several steps. The current node is identified in the first step, which becomes the start state of the path in the first iteration.

After that, the current node neighbors are found by calling Algorithm (3.5). In this algorithm, the current node neighbors are brought level by level starting with the level that has order 1. After that, each neighbor check whether they lie outside the robot environment boundaries or its status is obstacle to or dynamic obstacle to drop it from the current node neighbors. Then, if this neighbor has luciferin value greater than its own node (current node) luciferin value, it is dropped from the neighbors' list. This process is repeated until the number of accepted neighbors becomes equal to (nt) or the order of its level is equal to MVR value. Algorithm (3.5) achieves this task.

```
Algorithm (3.5): Find Current Node Neighbors

Input : the current node(X)
                constant Maximum Level (ML=3)
                Constant maximum number of current node
                neighbors (nt=10) that are located in the level
                that has order<= ML and their luc value < luc
                value of the current node.
      Output: current node neighbors
Begin
        1.for all X's neighbor node Y starting with the level
        of the first order
1.1 Check if (Y in the range of environment)
1.2Check if( Y within the ML)
                        1.2.1 check if the value of
                        Y.luc<X.luc Then make Y as
                        accepted neighbor to X
        3. If the number of the accepted neighbors of X<nt
        or the order of the neighbor node level<ML go to
        step 1
End Algorithm
```

Algorithm (3.5) to find Current Node Neighbors

The reduced set of neighbor nodes is obtained by calling the Algorithm (3.5). Algorithm (3.6) is used to determine the next current node that the mobile robot navigates to it. This Algorithm compares the accepted neighbors based on their probabilities to determine which node with the highest probability is the next.

```
Algorithm (3.6): Find Next Current Node
Input: The current node(X) and its Accepted node that are
determined by using Algorithm 3.7
output: The next current node
Begin
1. For each accepted neighbor node(Y) of the current node(X)
Compute the probability value(P) for the current node.
2- Y.P= (Y.luc – X.luc)/($\sum_{K=1}^{n}(K.luc - X.luc)$
3- Set the accepted node that has highest value of P to be
the Next Current node at the next iteration.
End Algorithm
```

Algorithm (3.6) to find Next Current Node

The obstacles are important to methods of path planning. In the robot environment, obstacles are divided into those with information known (static) and those with information unknown(dynamic).

The next section produces two algorithms; Algorithm (3.7) and Algorithm (3.8) to create and update a dynamic obstacle in the robot environment.

```
Algorithm (3.7): Crate Dynamic Obstacle
Input: Random selecting node(X1)
Temporal value of (X1)luciferin value(X.luc)
Output: Dynamic obstacle node(X1)
Begin
1.      Check if (( X1 in range of environment) and (status state is
clear))
1.1.Give X1 Deeppink color
   setpixel (X1,Deeppink);
        1.2.pixel[X1] = getpixel( X1);
         1.3. Make the Status of X1 as Dynamic obstacle
X1.status = " D_Obstacle ";
1.4.     Temporal.luc= X1.luc;
          1.5. X1.h= 10000;        // it is represent a very large value
    2. when the random dynamic obstacle selected in not clear   position
            2.1  Find Neighbor of x1                  % call Algorithm(3.7)
             2.2.for each neighbor Y1 of X1 do
                    2.2.1. check if (( Y1 in range of environment) and (status
state Y1 is clear)
                               2.2.1.1.setpixel (Y1,Deeppink);
                                2.2.1.2.pixel[Y] = getpixel(Y1);
                                2.2.1.3.Y1.status = "D_Obstacle";
                                2.2.1.4.temporal.h = Y1.h;
2.2.1.5.Y1.h= 10000;

End Algorithm
```

Algorithm (3.7) to create Dynamic Obstacle

Algorithm (3.7) is designed to create a dynamic obstacle by selecting its position in the clear space environment. Algorithm (3.8) moves the dynamic obstacle in a random position of a neighbor clear space environment.

```
Algorithm (3.8): Update Dynamic Obstacle
Input: Random selecting node(X1)
       Temporal value of (X1)luciferin value(X.luc)
Output: Dynamic obstacle node(X1)
begin
   1. X1.status = "Clear";
   2. X1.luc = temporal.luc ;
   3. setpixel (X1, White);
   4. Find Neighbors ( X1);          % call Algorithm(3.5)
   5. for each neighbor Y1 of X1 do
        5.1.  get the random position Y1 of neighbor
        5.2 check if ((Y1 in range of environment) and
                          (status state Y1 is clear)
            5.2.1 setpixel (Y1,DeepPink);
            5.2.2.pixel[Y1] = getpixel( Y1);
            5.2.3.Y1.status = "D_Obstacle";
            5.2.4.temporal.h = Y1.h;
            5.2.5.Y1.h= 10000;     // it is represent a very
                                            large value
            5.2.6.X1 =Y1 ;
End Algorithm
```

Algorithm (3.8) to update the dynamic obstacle

Finally, by calling the four previous Algorithms with the start state until reaching the goal state, the Algorithm (3.9) to move the mobile robot by Hybrid AD representation with the GSO algorithm is achieved.

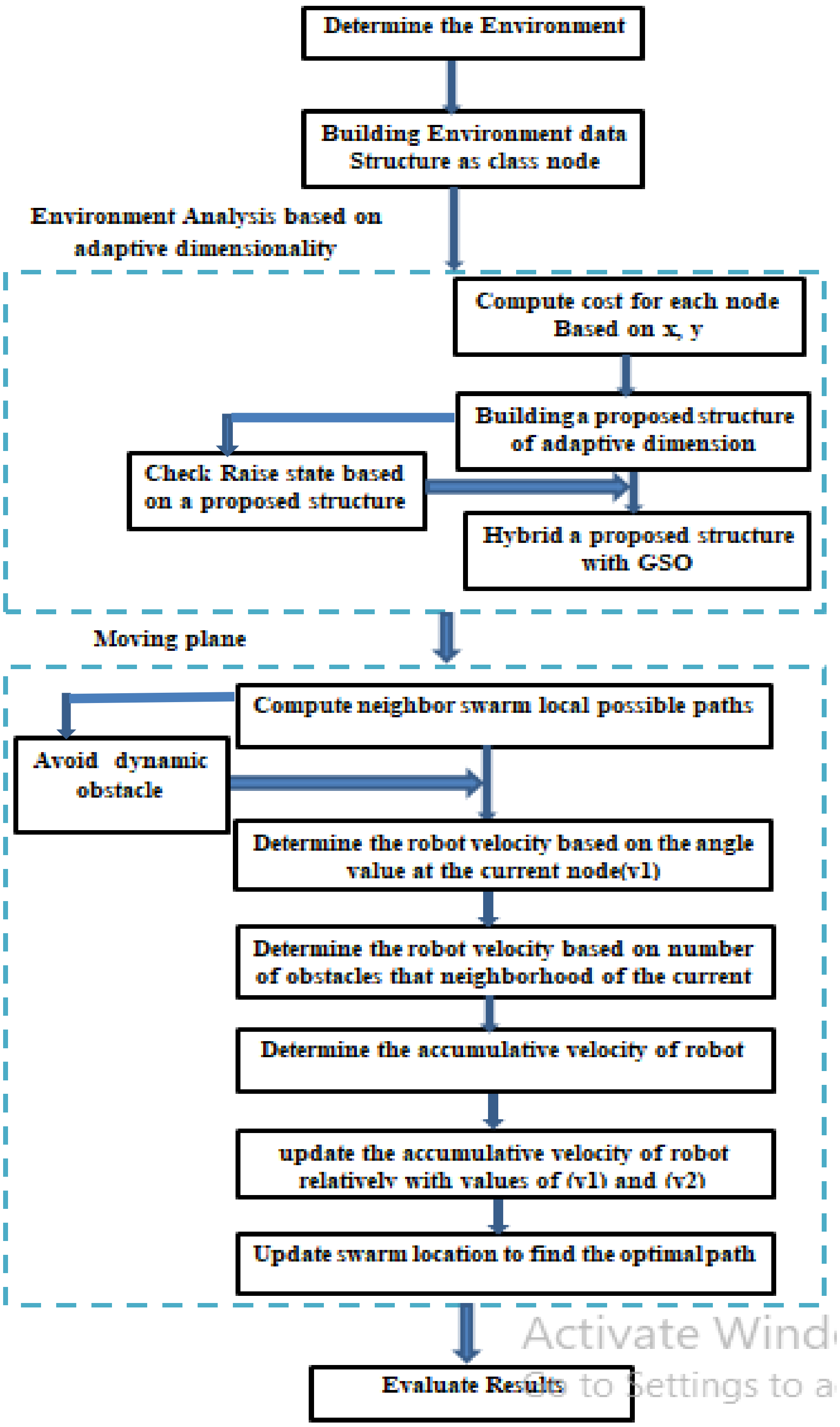
```
Algorithm (3.9):  Move Robot by Hybrid GSO

Input: Start state, Goal state.

Output: Robot path.

Begin

//Starting with start node to find the optimal path planning as current
node X.

1- Make start nod as the first current node.
2- Find the neighbors of the current node   % call Algorithm(3.5)
3- Find the next current node(Y)             % call Algorithm(3.6)
4- Let X=Y
5- Identify the sub path between X and Y based on the back index
and the value of luciferin.
6- Crate the dynamic obstacle                % call Algorithm(3.8)
7- Update the dynamic obstacle               % call Algorithm(3.10)
8- If ( Y≠ Goal state && iter-no< iter-max) go to step 1
End Algorithm
```

Algorithm (3.9) to move the Robot by Hybrid GSO with AD

3.3 The second improvement approach.

The second proposed approach intends to adapt the velocity of a robot that moves on a planned path by hybrid adaptive dimensionality with GSO algorithm. In this approach, the improvement focuses on the robot movement phase that was illustrated in the previous development proposed approach.

The improving process is achieved to adapt mobile robot velocity based on the impact of the two following factors: path rotation angle at each path node, and the number of obstacles that surround this node within its two closest levels towards the robot motion. On the other side, four new steps are added to the robot movement phase to adapt robot velocity. Therefore, many algorithms are designed in order to translate these additional steps. The following parts present these algorithms. Figure (3.2), illustrates second development approach’s phases.



Determine the Environment
Building Environment data Structure as class node
Environment Analysis based on adaptive dimensionality
Compute cost for each node Based on x, y
Building a proposed structure of adaptive dimension
Check Raise state based on a proposed structure
Hybrid a proposed structure with GSO
Moving plane
Compute neighbor swarm local possible paths
Avoid dynamic obstacle
Determine the robot velocity based on the angle value at the current node(v1)
Determine the robot velocity based on number of obstacles that neighborhood of the current
Determine the accumulative velocity of robot
update the accumulative velocity of robot relatively with values of (v1) and (v2)
Update swarm location to find the optimal path
Evaluate Results

Figure (3.2) second development approach's phases.

3.3.1 Compute the robot velocity

Before computing the robot velocity, there is a number of assumptions needed to achieve the simulation of results in the next chapter. Table (3.1) illustrates these assumptions of some robot attributes.

Table(3.1) assumptions of some robot attributes.

Robot attribute	value
Lowest velocity	20 velocity unit
Maximum velocity	80 velocity unit
Maximum velocity at angle 90	20 velocity unit
Maximum velocity at angle 45	60 velocity unit
Maximum velocity at angle 180	80 velocity unit
Acceleration value in horizontal and vertical direction	20 acceleration unit
Acceleration value in horizontal and diagonal direction	28 acceleration unit
Maximum deceleration	40 velocity unit

Algorithm (3.12) is used to find the cumulative robot velocity at each sub path node identified at each iteration which lies between the current node and the next path node. These sub optimal path of nodes is determined in the first step of the movement plan phase. Where, after determining the direction of robot motion to the next path node to identify the robot acceleration value 20 or 28. The allowed robot velocity at the next path node is computed based on the Equation (3.1).

Y.RobotVel = X.RobotVel+1/2*Acc*time ……(3.1).

Where, Y.RobotVel represents the cumulative robot velocity at the next node, while X.RobotVel is the robot velocity at the current path node, time represents the time of each transfer between the sub optimal two path nodes, allowed robot acceleration.

```
Algorithm (3.10): Find Robot Velocity
Inputs: velocity of robot acceleration(Vacc) which represent one of
    robot properties(40 velocity unite).
Distance(Dis) between the current node(X) and the next current
    node(Y).
The number of transfers(Tno) between nodes, here always equal 1.
Output: Robot velocity at each neighbor node.
Begin
1. Check the position of Y relative to X
2.  Compute the Robot acceleration
3. Compute the robot velocity at the next current node
   Y. RobotVel= X.vel + 1/2*Acc*time
End Algorithm
```

Algorithm (3.10) to find the Robot Velocity

3.3.2 Determine the robot velocity based on the number of surrounding obstacles

Algorithm (3.11) is designed to calculate the velocity of the robot at each robot path node. This is based on the number of obstacles that surround it within its two closest levels and in the direction of the robot's movement (Y.ObstVel**)**. In this algorithm, the robot velocity is based on the number of obstacles is determined. This is done by decreasing maximum robot velocity by Decreased Velocity(DecVel) per each obstacle neighbor of the next current node using Equation (3.2).

Y.ObstVel= MaxRobotVel - DecVel ………….. (3.2)

Where: Y.ObstVel represents the robot velocity at the next path node based on the number of obstacles that lie in its closest two levels, while Maximum Robot Velocity(MaxRobotVel) is one of robot attributes. DecVel represents the value of decreased velocity per each obstacle (6 velocity units if the obstacle lies in the first level, while 3 velocity unit if it locates in the second level of X) which is one of robot attributes.

```
Algorithm (3.11): Find Robot ObstVel

Input:  the neighbor nodes of the current node, the maximum robot
        velocity(MaxRobotVel) and  the value of decreased velocity
        per each  obstacle(DecVel1)
Output: the Robot ObstVel for each neighbor node.

Begin
    1.For( each Y neighbor to current node(X) in two its closest
    levels)
       1.1.Check If Y  is obstacle
          1.1.1. Check the order of obstacle level
          1.1.2. Y.RobotVel= MaxRobotVel - DecVel
End Algorithm
```

Algorithm (3.11) to find Robot Obstacle Velocity

3.3.3 Compute the robot velocity based on the rotation angle value at the current path node

After determining the velocity of the mobile robot at each optimal path node based on the number of obstacles that are surrounding it within its two closest levels, the rotation angle at each path node needs identification. Robot velocity based on the value of this angle is calculated because it is one of the important and impacting factors in identifying the suitable mobile robot velocity. To achieve all these previous steps, Algorithm (3.12) is designed.

Algorithm (3.12): Find Robot Angle Velocity

Input: The maximum robot velocity(MaxRobotVel) which is as one of robot attributes.

Output: Robot velocity at the current node based on its surrounding obstacle at the two closest levels(RobotAngleVel)

```
// starting from the second path node.
Begin
  1. For( each path node)
       1.1.Calculate the Value of RotationAngle(AngleVel) at
           the current node based on the locations of the previous
           node(F), current node(X) and next node(Y);
       1.2.identify the value of RobotAngleVel at X based on the
           Robot attributes
End Algorithm
```

Algorithm (3.12) to find the Robot Angle Velocity

3.3.4 Moving the mobile robot with an adaptive velocity

Algorithm (3.13) is designed to move the mobile robot from the start state to the goal state of the optimal path with the adaptive velocity at each optimal path node based on the number of obstacles that are surrounding it within its two closest levels, and the rotation angle at each path node. This algorithm starts with the start state of the robot path as a current node. The neighbor nodes of the current node is identified in the next step of the Algorithm by calling Algorithm (3.1). In the next step of this algorithm, the next node is determined by calling Algorithm (3.2).

Then, the dynamic obstacles are created by calling algorithm (3.3) to Create Dynamic Obstacle randomly at each iteration, While in the next step, the location of the dynamic obstacle is updated randomly by using Algorithm (3.4) to Update Dynamic Obstacle.

Three algorithms are called, Algorithm (3.6), Algorithm (3.7), and Algorithm (3.8), to adapt the robot velocity at each optimal path node. These steps are repeated till the stop condition is achieved. The stop condition is fulfilled when one of the two cases occurs. Either the number of iterations (iter-no) becomes equal to the maximum number of iterations (iter-max), or the next current path node becomes the Goal state.

The last phase of the second approach diagram is the evaluation results phase. In this phase, the improvement approach is applied to adapt the robot velocity on several environment paths that are different in the number of surrounding obstacles and the value of rotation angle at each path node. After that, the obtained results improvements approach are compare to the results of another approach to evaluate the efficiency of adaptive robot velocity approach of this dissertation.

```
Algorithm( 3.13):  Move Robot with adaptive velocity
Input: position of the start state
Output: adaptive robot velocity at each planned path node.
Begin
1-  Make start nod as the first current node.

2-  Find the neighbors of the current node   % call Algorithm(3.5)

3-  Find the next current node(Y)            % call Algorithm(3.6)

4-  Let X=Y

4-   Identify the sub path between X and Y based on the back index
     and the value of luciferin.
5-  Crate the dynamic obstacle               % call Algorithm(3.7)
6-  Update the dynamic obstacle              % call Algorithm(3.8)
7-  Compute the value of RobotVel            %call Algorithm (3.11)
8-  Compute the value of RobotObstVel        %call Algorithm (3.12)
9-  Compare the RobotVel with RobotObstVel and set
      the minimum one as RobotVel
10- Compute the value of RobotAngleVel   %call Algorithm (3.14)
11- Compare the RobotVel with RobotAngleVel and set the
     minimum one as RobotVel
12- If ( Y≠ Goal state && iter-no< iter-max) go to step 1

End Algorithm
```

Algorithm(3.13) to move the robot with adaptive velocity

Chapter Four

SIMULATION AND RESULTS

4.1 Introduction

This chapter will present a new proposed simulation Application for Swarm Robot Path Planning, which is tailored specially to move robot path planning in a dynamic environment. This application will simulate the implementation of the two proposed approaches of this research study on several dynamic environments that are different in their complexity and variant number of gate raise state. Using C# Microsoft Visual Studio 2017 on a core i7 PC, the simulation results of the two improvement approaches of this research study are implemented. The simulation and the results evaluation is divided on to two parts, one part for each improvement approach of this study. Also, it should be noted that two types of environments will be used, which differ in terms of their sizes, 10x10 and 50x50.

In the first part the proposed simulation application simulation is used to obtain the results and find the optimal path with the path planning based on hybrid AD with the GSO approach. While in the second, part the proposed simulation application simulation is used to obtain the results of adaptive robot velocity.

4.2 The first improvement hybrid AD with GSO illustrative example

The following part produces an example that shows how the proposed improvement approaches operate. This example is implemented to present results in detail in order to find the optimal path.

In Figure (4.1), small environment with size (10*10) is illustrated and which will use in the guide example. Static obstacles in this environment shown in black. While, the selecting start state(S) is marked by the red at the location (0,9) and goal state(G) marked by the gold at (7,0).

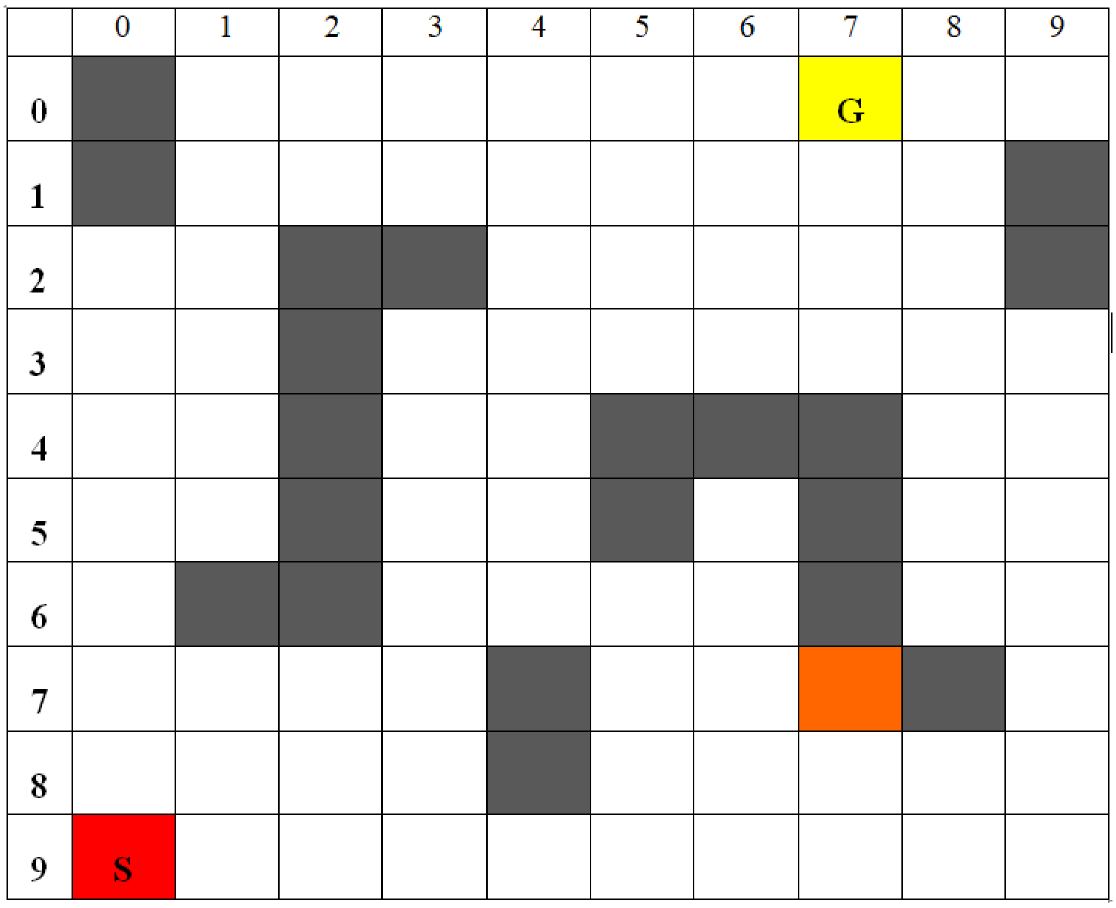


Figure (4.1) Small environment used for guide example

The robotic environment needs to be selected firstly from the library storage in order to find the optimal path by using a hybrid AD representation with the GSO approach. This environment selection process is achieved using an open dialog box window that chooses drives, directory and the image environment. Then, it will be displayed on the screen. Figure (4.1) illustrates this robotic environment.

After that, selecting of the start and goal states is achieved. Then, every cell in the robot environment is read and then stored in the array of class nodes. The class array was defined as a class data structure. This class name node includes, number data members; status, tag, X-coordinate of the node(ni), Y-coordinate of the node(nj), arc cost(luc), smallest arc cost(k).

At the beginning, all the cells of the robotic environment are defined as new, then Start, Goal, Clear and Obstacle are given as cases of the status states. On the analysis and compute cost stage, by D* algorithm an open list of states is kept which is used to propagate the information changes to the values of arc cost and to compute states path costs in the space. At the beginning, the current node is a state of goal (7,0), which is inserted in the queue OPEN list. After that, by finding its 8 neighborhood states, this current node will be expanded as shown Figure (4.2). Each one of these neighbors must be located inside the environment and in the states of the current node that Their status is free clear ((6,0), (6,1), (7,1), (8,0), and (8,1)). Figure (4.2) illustrates the Eight neighbor direction states.

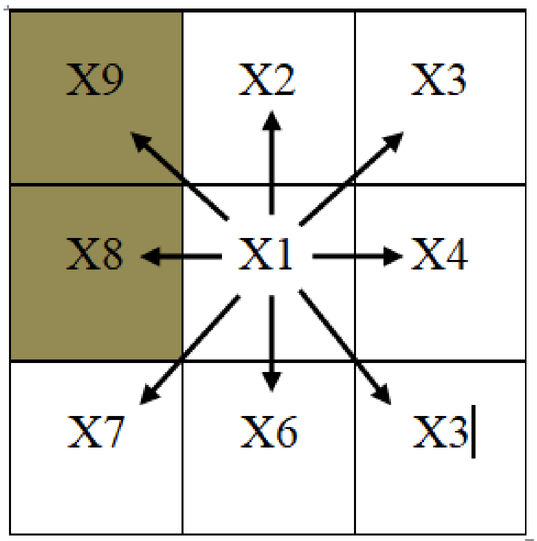
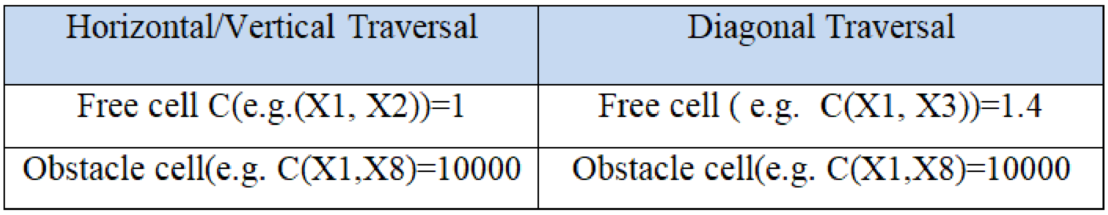


Figure (4.2) Eight neighbor direction states

The calculating the cost of the arc for each one of the neighbors of the current node is done by giving a high cost if it is an obstacle (10000) (for example (1,8), while if the state is a new sign and a clear status, it must first check its location compared to the current node then, add 1.4 to the arc cost if it lays on diagonal position else add 1 only. Table (4.1) illustrates that.

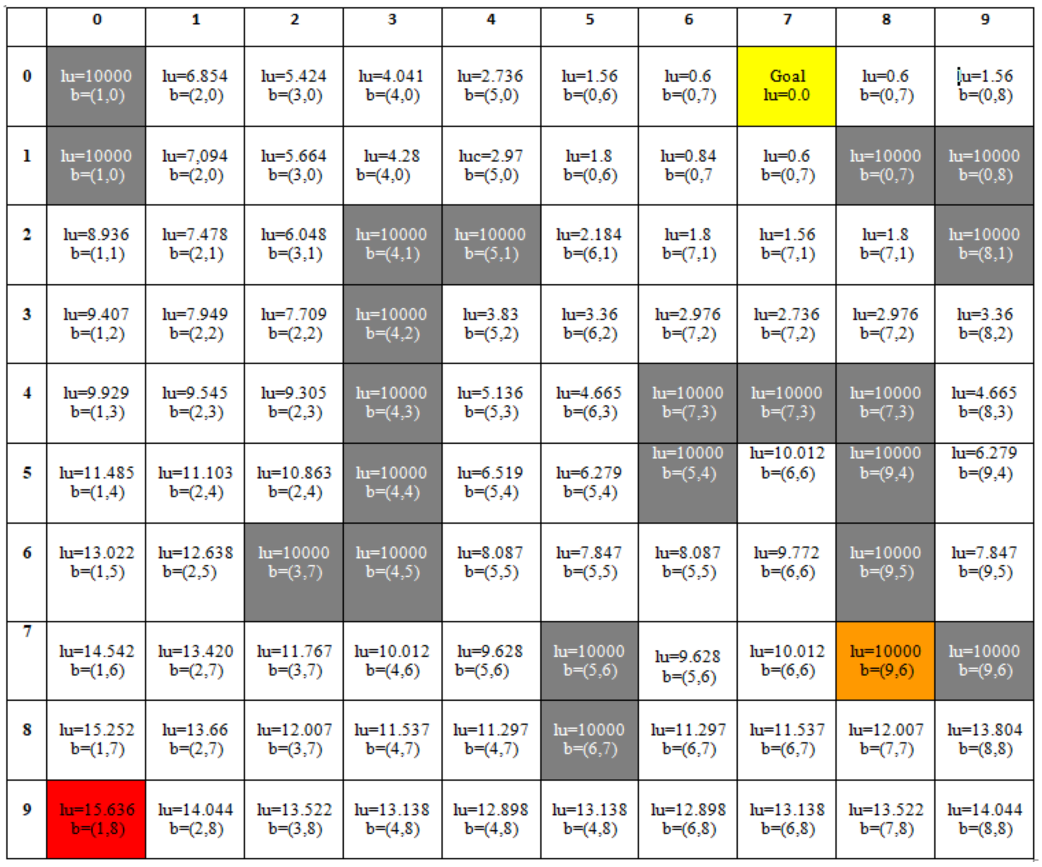
Table (4.1) Computed arc cost to all possible cases of free and obstacle status



Horizontal/Vertical Traversal	Diagonal Traversal
Free cell C(e.g.(X1, X2))=1	Free cell (e.g. C(X1, X3))=1.4
Obstacle cell(e.g. C(X1,X8)=10000	Obstacle cell(e.g. C(X1,X8)=10000

Table (4.1) illustrates the values of arc cost and back pointer of environment in the previous example that are found starting from state of goal and repeatedly expanded until they reach state of start.

Table (4.2) The arc cost values and back pointer states in first improvement approach.



	0	1	2	3	4	5	6	7	8	9
0	lu=10000 b=(1,0)	lu=6.854 b=(2,0)	lu=5.424 b=(3,0)	lu=4.041 b=(4,0)	lu=2.736 b=(5,0)	lu=1.56 b=(0,6)	lu=0.6 b=(0,7)	Goal lu=0.0	lu=0.6 b=(0,7)	lu=1.56 b=(0,8)
1	lu=10000 b=(1,0)	lu=7,094 b=(2,0)	lu=5.664 b=(3,0)	lu=4.28 b=(4,0)	luc=2.97 b=(5,0)	lu=1.8 b=(0,6)	lu=0.84 b=(0,7	lu=0.6 b=(0,7)	lu=10000 b=(0,7)	lu=10000 b=(0,8)
2	lu=8.936 b=(1,1)	lu=7.478 b=(2,1)	lu=6.048 b=(3,1)	lu=10000 b=(4,1)	lu=10000 b=(5,1)	lu=2.184 b=(6,1)	lu=1.8 b=(7,1)	lu=1.56 b=(7,1)	lu=1.8 b=(7,1)	lu=10000 b=(8,1)
3	lu=9.407 b=(1,2)	lu=7.949 b=(2,2)	lu=7.709 b=(2,2)	lu=10000 b=(4,2)	lu=3.83 b=(5,2)	lu=3.36 b=(6,2)	lu=2.976 b=(7,2)	lu=2.736 b=(7,2)	lu=2.976 b=(7,2)	lu=3.36 b=(8,2)
4	lu=9.929 b=(1,3)	lu=9.545 b=(2,3)	lu=9.305 b=(2,3)	lu=10000 b=(4,3)	lu=5.136 b=(5,3)	lu=4.665 b=(6,3)	lu=10000 b=(7,3)	lu=10000 b=(7,3)	lu=10000 b=(7,3)	lu=4.665 b=(8,3)
5	lu=11.485 b=(1,4)	lu=11.103 b=(2,4)	lu=10.863 b=(2,4)	lu=10000 b=(4,4)	lu=6.519 b=(5,4)	lu=6.279 b=(5,4)	lu=10000 b=(5,4)	lu=10.012 b=(6,6)	lu=10000 b=(9,4)	lu=6.279 b=(9,4)
6	lu=13.022 b=(1,5)	lu=12.638 b=(2,5)	lu=10000 b=(3,7)	lu=10000 b=(4,5)	lu=8.087 b=(5,5)	lu=7.847 b=(5,5)	lu=8.087 b=(5,5)	lu=9.772 b=(6,6)	lu=10000 b=(9,5)	lu=7.847 b=(9,5)
7	lu=14.542 b=(1,6)	lu=13.420 b=(2,7)	lu=11.767 b=(3,7)	lu=10.012 b=(4,6)	lu=9.628 b=(5,6)	lu=10000 b=(5,6)	lu=9.628 b=(5,6)	lu=10.012 b=(6,6)	lu=10000 b=(9,6)	lu=10000 b=(9,6)
8	lu=15.252 b=(1,7)	lu=13.66 b=(2,7)	lu=12.007 b=(3,7)	lu=11.537 b=(4,7)	lu=11.297 b=(4,7)	lu=10000 b=(6,7)	lu=11.297 b=(6,7)	lu=11.537 b=(6,7)	lu=12.007 b=(7,7)	lu=13.804 b=(8,8)
9	lu=15.636 b=(1,8)	lu=14.044 b=(2,8)	lu=13.522 b=(3,8)	lu=13.138 b=(4,8)	lu=12.898 b=(4,8)	lu=13.138 b=(4,8)	lu=12.898 b=(6,8)	lu=13.138 b=(6,8)	lu=13.522 b=(7,8)	lu=14.044 b=(8,8)

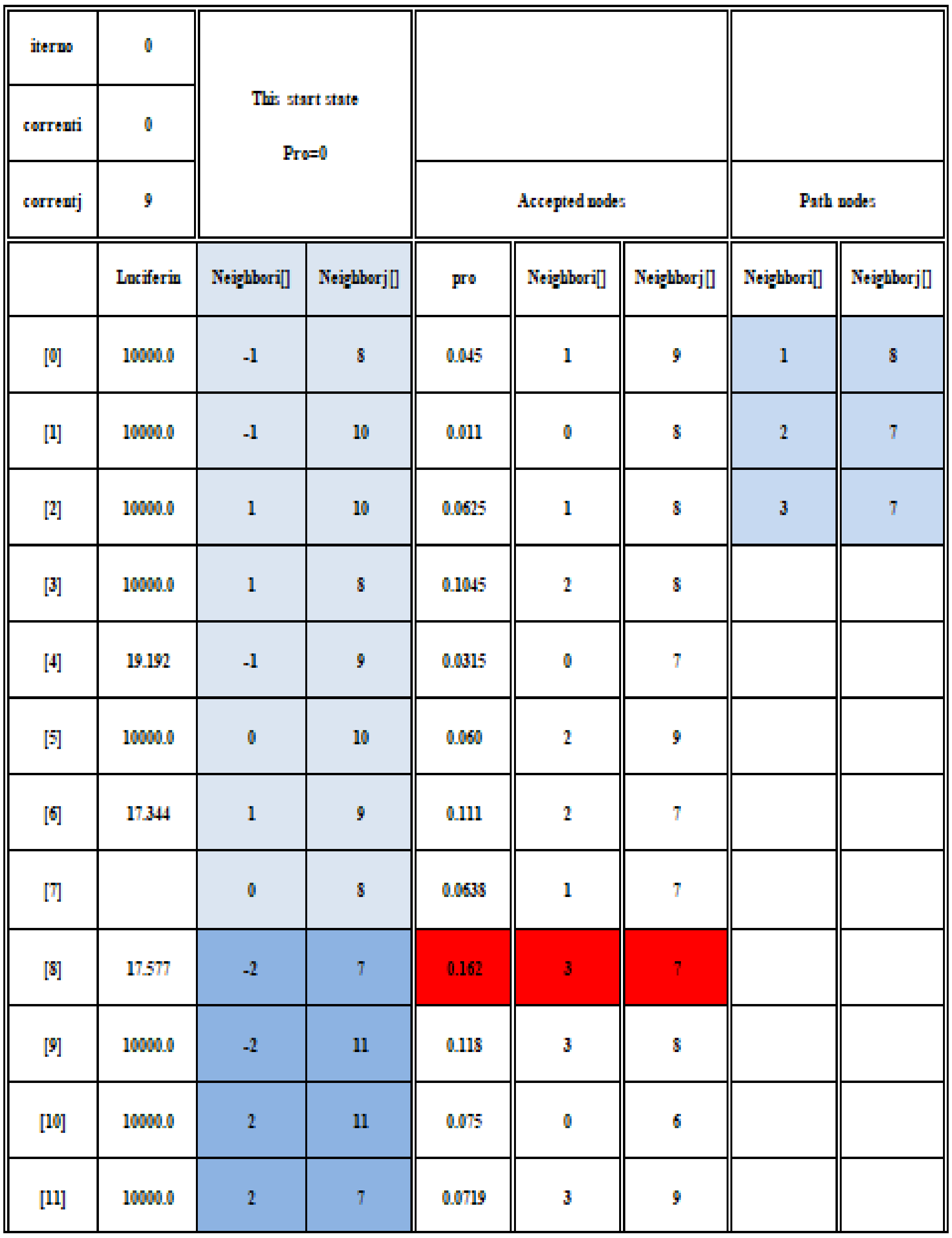
Finishing the phase of analysis and compute arc cost, provides to the robot all the known environment information. While, when the moved dynamic obstacles is created the unknown information of dynamic environment is coming from them (phase of avoiding dynamic obstacle).

Now, the move phase will be started to find the next specific solution using hybrid AD representation with GSO approach for moving from start state (0,9) to the next node. Implementing of this phase several includes several steps that are achieved by calling algorithms appropriate for them. the following parts illustrate that.

Firstly, at each iteration number(iterno), here at iterno (0), the accepted neighbors of the current node (start state) is found by calling Find Current Node Neighbors algorithm. At first, the neighbors at first level will be checked to drop each neighbor that is outside the robot work environment and also that its luciferin value(luc) more than the luciferin value of the current node to find the accepted neighbor nodes (the rest nodes). if the stop condition is not satisfied, same steps will repeat on the neighbors at second level and third level else what was obtained from the neighbors will be enough to start with the step of selecting the next current node to move forward to it.

Then, the next current node is selected by calling Find Next Current Node Algorithm to select it based on the values of accepted nodes probabilities that were obtained previously. The next current node is the accepted node that has highest probability value. After that the sub optimal path will be found by determining the intermediate nodes between the current node and the next current node based on the back pointer value, the luciferin value and the connections between the accepted neighbor nodes. Now, before the next iteration is started the next current node (7,3) It should be instead of (0,9) in order to become as current node in the next iteration (iterno 1). Table (4.3) illustrates the selection the next current node and determine the sub optimal path at iteration 0.

Table (4.3) selection the next current node and determine the sub optimal path at iteration 0.

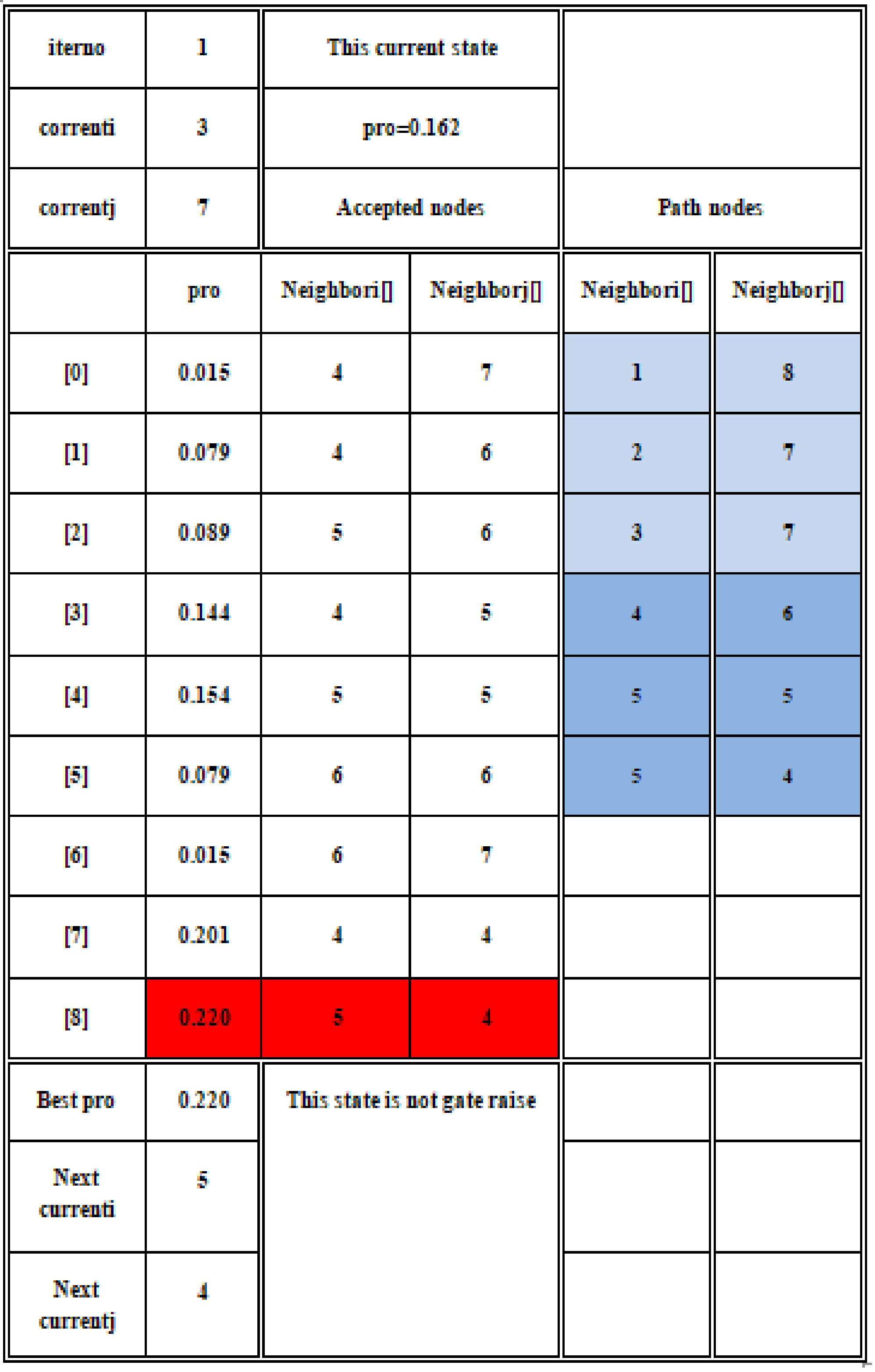


iterno	0	This start state Pro=0						
correnti	0							
correntj	9			Accepted nodes			Path nodes	
	Luciferin	Neighbori[]	Neighborj[]	pro	Neighbori[]	Neighborj[]	Neighbori[]	Neighborj[]
[0]	10000.0	-1	8	0.045	1	9	1	8
[1]	10000.0	-1	10	0.011	0	8	2	7
[2]	10000.0	1	10	0.0625	1	8	3	7
[3]	10000.0	1	8	0.1045	2	8		
[4]	19.192	-1	9	0.0315	0	7		
[5]	10000.0	0	10	0.060	2	9		
[6]	17.344	1	9	0.111	2	7		
[7]		0	8	0.0638	1	7		
[8]	17.577	-2	7	0.162	3	7		
[9]	10000.0	-2	11	0.118	3	8		
[10]	10000.0	2	11	0.075	0	6		
[11]	10000.0	2	7	0.0719	3	9		

[12]	10000.0	-2	8	0.0863	1	6		
[13]	10000.0	-1	11					
[14]	10000.0	2	10					
[15]	10000.0	1	7					
[16]	10000.0	-2	9					
[17]	18.039	0	11					
[18]	10000.0	2	9					
[19]	18.279	0	7					
[20]	10000.0	-2	10					
[21]	10000.0	1	11					
[22]	15.695	2	8					
[23]	15.935	-1	7					
[24]	17.721	-3	6					
[25]	10000.0	-3	12					
[26]	10000.0	3	12					
[27]	10000.0	3	6					
[28	10000.0	-3	7					
[29]	10000.0	-2	12					
[30]	10000.0	3	11					
[31]	10000.0	2	6					

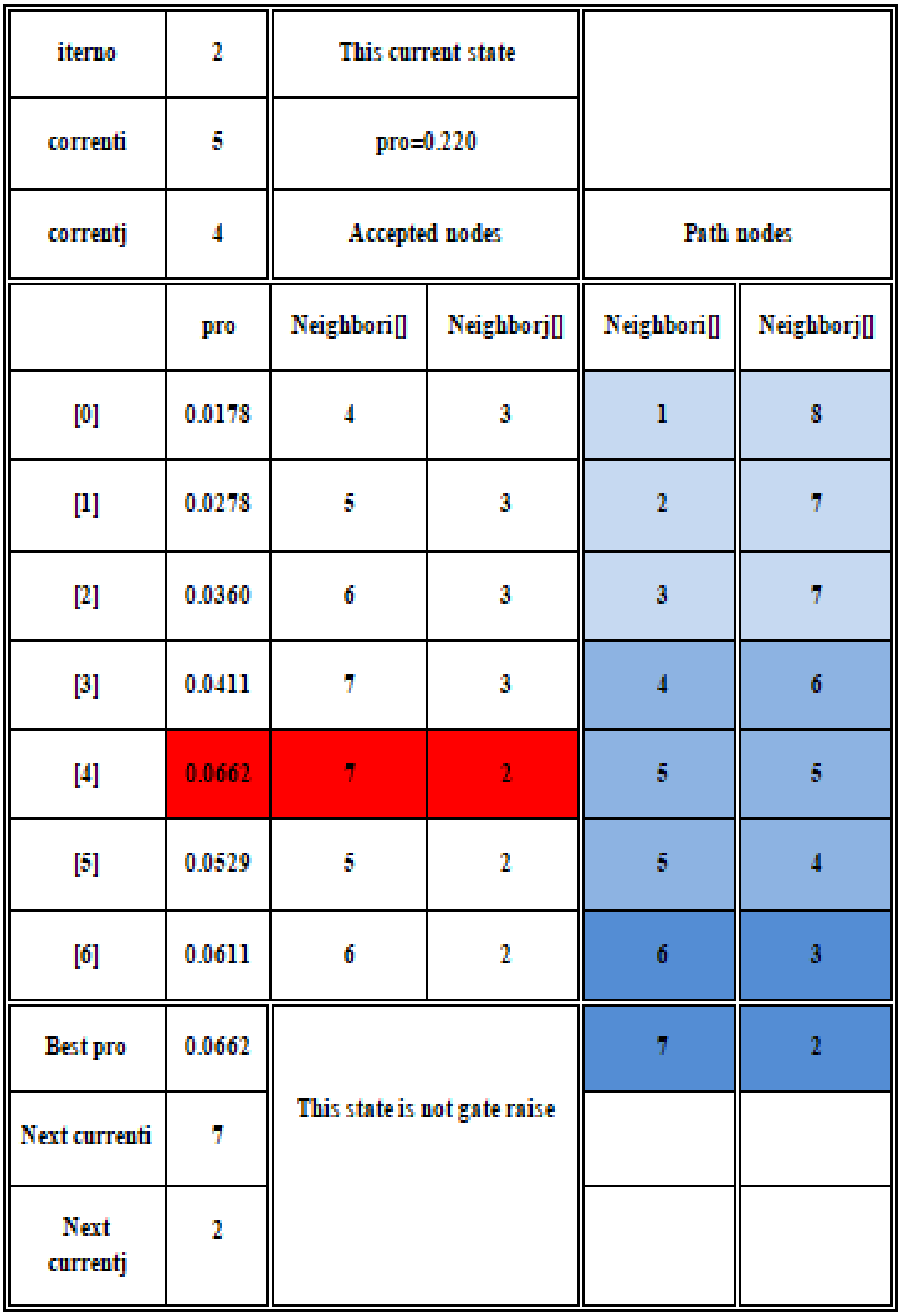
[32]	10000.0	-3	8					
[33]	10000.0	-1	12					
[34]	10000.0	3	10					
[35]	10000.0	1	6					
[36]	16.465	-3	9					
[37]	10000.0	0	12					
[38]	16.465	3	9					
[39]	10000.0	0	6					
[40]	10.012	-3	10					
[41]	10000.0	1	12					
[42]	10000.0	3	8					
[43]	10000.0	-1	6					
[44]	10000.0	-3	11					
[45]	13.959	2	12					
[46]	10000.0	3	7					
[47]	10000.0	-2	6					
Best pro	0.162							
Next currenti	3							
Next currentj	7							

Table (4.4) illustrates the selection the next current node and determine the sub optimal path at iteration 1.



iterno	1	This current state			
correnti	3	pro=0.162			
correntj	7	Accepted nodes		Path nodes	
	pro	Neighbori[]	Neighborj[]	Neighbori[]	Neighborj[]
[0]	0.015	4	7	1	8
[1]	0.079	4	6	2	7
[2]	0.089	5	6	3	7
[3]	0.144	4	5	4	6
[4]	0.154	5	5	5	5
[5]	0.079	6	6	5	4
[6]	0.015	6	7		
[7]	0.201	4	4		
[8]	0.220	5	4		
Best pro	0.220	This state is not gate raise			
Next currenti	5				
Next currentj	4				

Table (4.5) illustrates the selection the next current node and determine the sub optimal path at iteration 2.



iterno	2	This current state			
correnti	5	pro=0.220			
correntj	4	Accepted nodes		Path nodes	
	pro	Neighbori[]	Neighborj[]	Neighbori[]	Neighborj[]
[0]	0.0178	4	3	1	8
[1]	0.0278	5	3	2	7
[2]	0.0360	6	3	3	7
[3]	0.0411	7	3	4	6
[4]	0.0662	7	2	5	5
[5]	0.0529	5	2	5	4
[6]	0.0611	6	2	6	3
Best pro	0.0662	This state is not gate raise		7	2
Next currenti	7				
Next currentj	2				

Table (4.6) illustrates the selection the next current node and determine the sub optimal path at iteration 3.

iterno	3	This current state			
correnti	7	pro=0.0662			
correntj	2	Accepted nodes		Path nodes	
	pro	Neighbori[]	Neighborj[]	Neighbori[]	Neighborj[]
[0]	0.08	6	1	1	8
[1]	0.106	7	1	2	7
[2]	0.08	8	1	3	7
[3]	0	5	0	4	6
[4]	0.106	6	0	5	5
[5]	0	9	0	5	4
[6]	0.173	7	0	6	3
[7]	0.106	8	0	7	2
Best pro	0.0662	Goal state and This state is not gate raise		7	1
Next currenti	7			7	0
Next currentj	0				

The experimental results of the previous example guide that has one state of gate rise to find the optimal path in in dynamic environment. It includes 4 iterations, the total path arc cost is 13.4, and time occupy is 64 sec.

4.3 Avoid collision Dynamic Obstacle

In dynamic environments, overcoming collision with dynamic obstacle is a key function, which will be presented in this section. An implementation of experimental based on the same example guide environment in which the robot is intersection with the dynamic obstacle at the same location and how the robot locally deals with to overcome the collision with it instantly**.**

Overcoming the obstacles that will fall on the path of the mobile robot using proposal improvement hybrid AD with GSO will be tested using the first experiment. In this experiment, the robot will move from the start state to find its optimal path. But when it arrives to position (7,1) which represents the best solution (next current node) in iteration 3. The mobile robot finds that this location is occupied by one of the dynamic obstacles. Therefore, the robot changes the direction of its movement around the position (7,1) to avoid collision to find it as the best solution (next current node). so that, the total path cost became 14.8 instead of 13.4.

.

4.4 Comparison Results of the first improvement approach

In this part, a brief overview about the planning is presented to evaluate the efficiency of the AD-GSO approach. Therefore, the results that had been obtained by applying the proposed approach on the two different sizes of environments: 10x10 and 50x50 are compared. The hybrid adaptive dimensionality with GSO algorithm parameters used in the test for all the different environments as shown in Table(4.7).

Table (4.7) The Parameters Used in the Hybrid Adaptive Dimensionality with GSO Algorithm.

Parameter	**ρ**	**γ**	**nt**	**MVR**
Value	0.4	0.6	10	3

i. Select the number of accepted neighbour nodes (nt).

Many simulations are conducted for applicating the first improvement approach of this study to move the robot on various paths that lie on different 50x50 environments. In these simulations, it was taken into consideration that the number of accepted neighbor nodes of the current path node is different at each time (eg. 8, 9, 10, 11, 12). These simulations are proven not to accepts doubt that the best number of accepted neighbor nodes (nt) of the current node to select among them the next current node which the robot must move to it later is 10 nodes. The reason for that return to achieve it the best equilibrium between the number of iterations and the path cost for the planned path comparing with the other rest numbers. The results are obtained from these simulations which are illustrated in Table (4.8) and Table (4.9) show that.

Table (4.8) the results of simulation of applying the first improvement approach using different number of accepted neighbor nodes

Path no.	nt	Iter-no	Path-cost
1	**8**	**13**	**42.8**
	9	**13**	**41.8**
S(8,17)	**10**	**12**	**41.8**
	11	**12**	**41.8**
G(44,8)	**12**	**12**	**41.8**
2	**8**	**19**	**58.4**
	9	**18**	**58.4**
S(5,2)	**10**	**18**	**58.4**
	11	**18**	**58.4**
G(22,48)	**12**	**18**	**58.4**
3	**8**	**23**	**78.4**
	9	**22**	**77.2**
S(0,0)	**10**	**22**	**77.2**
	11	**22**	**77.2**
G(49,49)	**12**	**22**	**77.2**
4	**8**	**20**	**73.4**
	9	**20**	**72.8**
S(10,0)	**10**	**19**	**71.6**
	11	**18**	**72.4**
G(49,47)	**12**	**18**	**72.4**
5	**8**	**23**	**72.4**
	9	**22**	**72.4**
S(5,48)	**10**	**22**	**72.4**
	11	**21**	**72.4**
G(37,11)	**12**	**21**	**72.4**
6	**8**	**20**	**77.2**
	9	**19**	**77.8**
S(28,23)	**10**	**16**	**66.2**
	11	**16**	**66.2**
G(14,29)	**12**	**16**	**66.2**
7	**8**	**23**	**67.8**
	9	**22**	**67.8**
S(37,0)	**10**	**20**	**67.8**
	11	**20**	**67.8**
G(4,49)	**12**	**19**	**68.4**
8	**8**	**18**	**56.8**
	9	**17**	**57.6**
S(38,1)	**10**	**16**	**56.2**
	11	**16**	**56.2**

G(47,49)	12	16	56.2
9	8	19	72
	9	19	72.6
S(13,4)	10	18	72.2
	11	18	72.2
G(22,29)	12	18	72.2
10	8	17	57.2
	9	17	57.2
S(46,22)	10	16	57.2
	11	16	57.2
G(0,45)	12	16	57.2

Table(4.9) The average of iteration number and the average of path cost for each number of accepted neighbor nodes.

Accepted nodes (nt)	Average of Iteration number	Average of path cost
8	19.5	65.4
9	18.9	65.56
10	17.9	64.1
11	17.7	64.22
12	17.6	64.24

The simulation to validate the proposed approach is divided into the following two parts:

Part 1: The first part illustrates the simulation results of the proposed approach using a small chosen environment of $(10 * 10)$ pixel size which includes only two gate raise states. The example illustrates results to achieve this simulation using the start state at position (0, 9) and the destination position at (7, 0). The results of the simulation of the optimal path is clarified in the green color, the total cost equals 16.2, the time occupy equals 0.5098 sec and the number of the iterations equals 4.

The obtain results by application the Hybrid mixed D* with Lbest PSO, hybrid improved D* with Lbest PSO, and hybrid AD with GSO approaches on the same 10x10 environment size that is illustrated in figure (5,5), and also with the same start state (0,9) and goal state (7,0) illustrates the effectiveness of the proposed research approach to find the optimal path in less number of iterations (4) with a lower path cost (16. 2), and in a shorter time occupy (0.0598) compared with other approaches. As shown in the table (4.10).

Table (4.10) The comparison experiment simulation results of mixed D∗ with Lbest PSO, hybrid improved D* and LPSO and hybrid AD with GSO approaches.

Approach Name	**Environment Size**	**Start and Goal states**	**Iteration No.**	**Path cost**	**Time occupy**
Hybrid mixed D* with Lbest PSO	**10x10**	**S(0,9) G(0,7)**	**4**	**98.2**	**1.67**
Hybrid improved D* with Lbest PSO			**12**	**53.4**	**1.828**
Hybrid AD with GSO			**4**	**16.2**	**0.5098**

Part2. This part presents the simulation results of some comparison experiments of hybrid mixed D∗ algorithm with Lbest PSO, the hybrid improved D* algorithm with Lbest PSO approach on a dynamic environment, and the hybrid AD-GSO algorithm by using six different(50x50) cell environments as illustrated in figure (4.3).

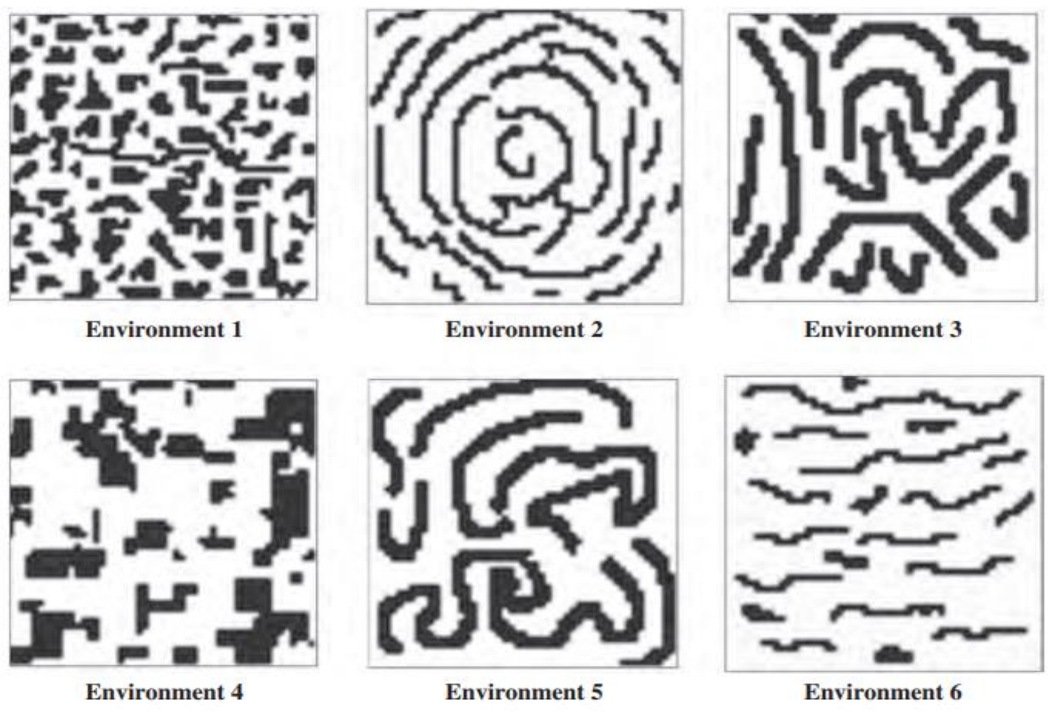


Figure (4.3) The six Different Environments of Size 50 ∗ 50 Cell Used in Comparison.

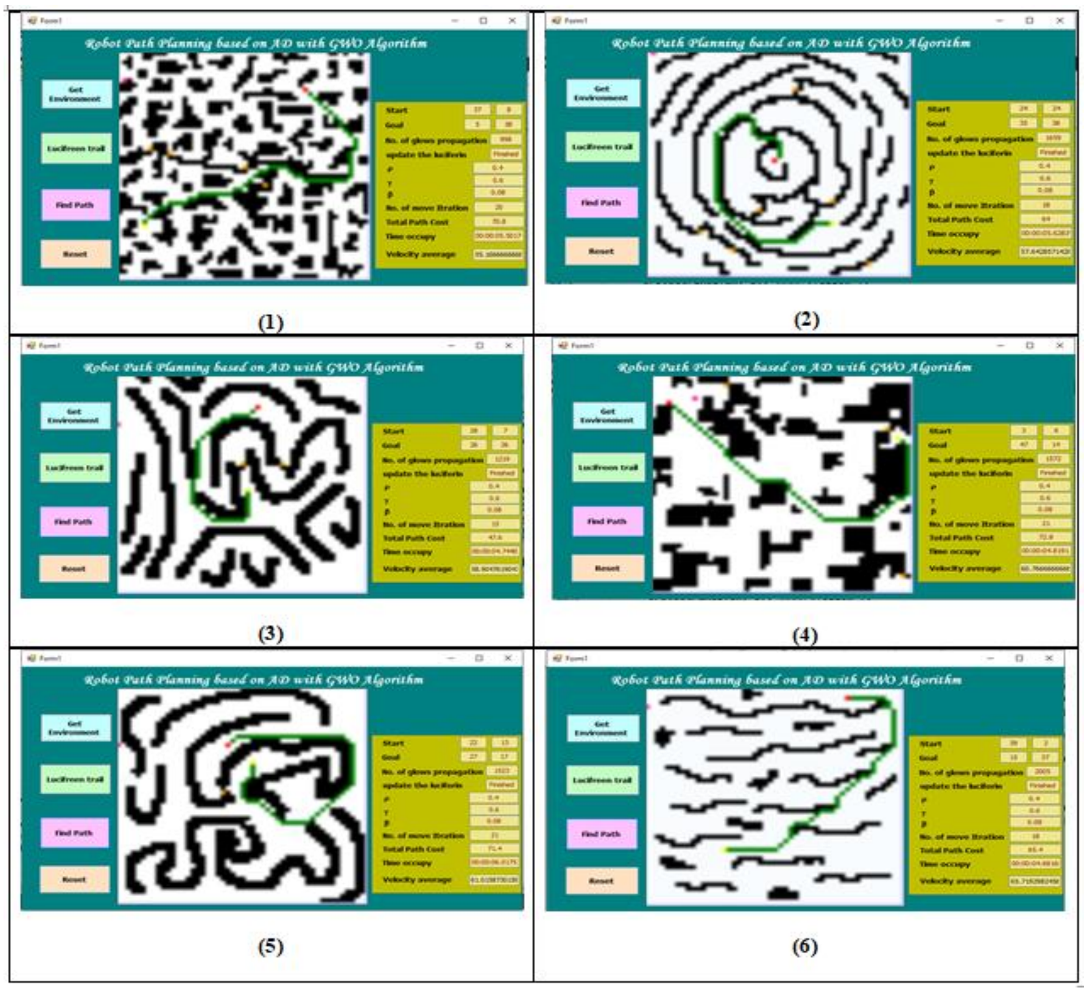
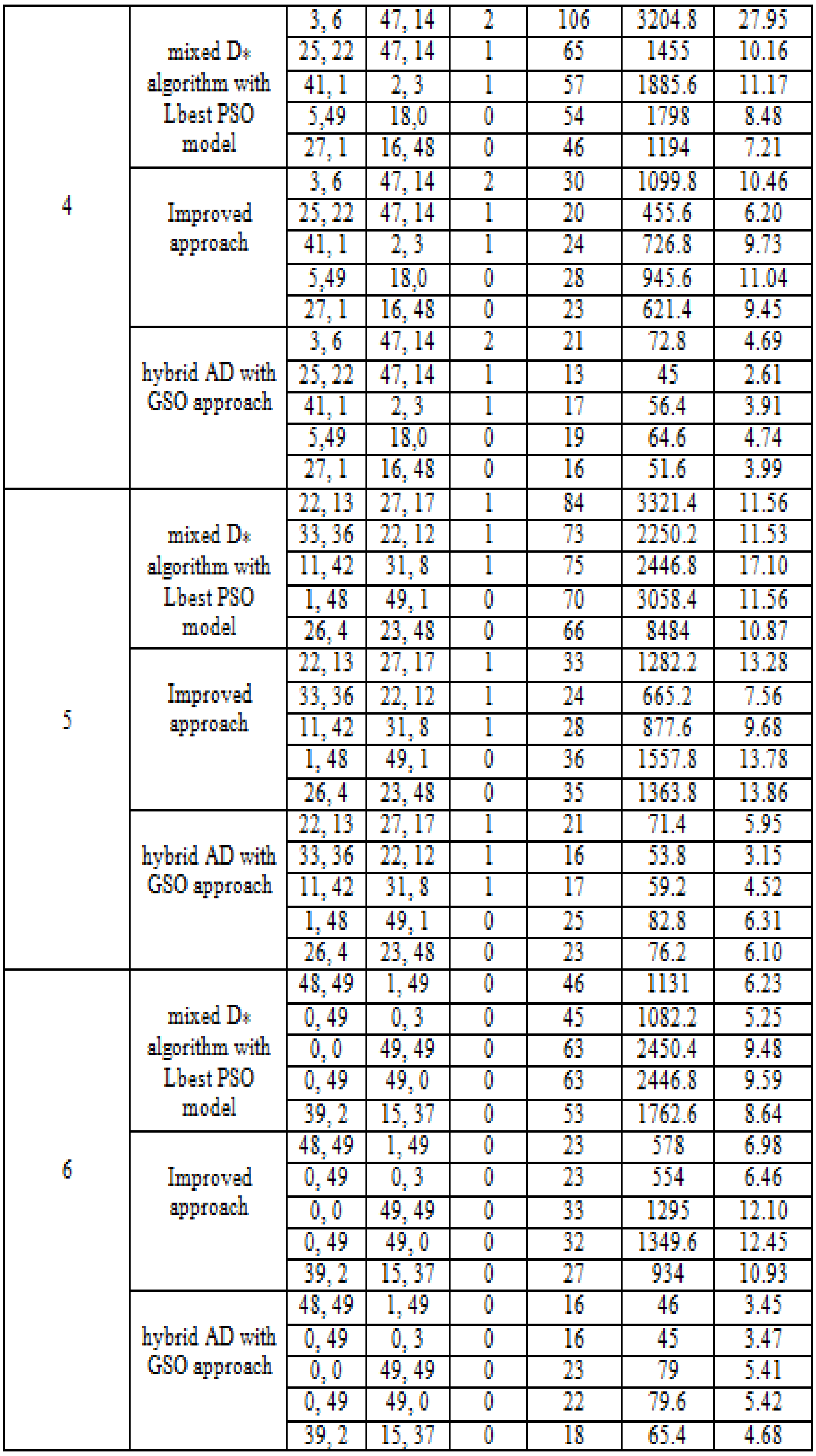


Figure (4.4) Finding the Robotic Path for Six Selected implementation on different five complex environments.

In the beginning of this part, the results of applying the proposed approach on each of the different six environments, illustrated in figure (4.4), will be achieved by selecting one case for each. The results of this simulation are displayed in Table (4-11).

Table (4-11) The comparison experiment simulation results on six variant environments

Environment No.	Algor.name	Start state	Goal state	No. of Gate	No. of iteration	Total Arc cost	Time occupy
1	mixed D* algorithm with Lbest PSO model	37, 8	5, 38	3	155	6440.2	2.34.76
		45, 6	19, 41	3	153	6656.8	1.22.65
		0, 42	6, 2	3	115	4357.4	1.19.04
		3,2	2, 48	3	127	5960.2	59.03
		21, 14	21, 37	3	115	4393	34.18
	improvement approach	37, 8	5, 38	3	32	1213.4	9.54
		45, 6	19, 41	3	26	827	8,13
		0, 42	6, 2	3	47	2633	15.42
		3,2	2, 48	3	50	2820.4	16.39
		21, 14	21, 37	3	28	929	9.45
	hybrid AD with GSO	37, 8	5, 38	3	20	70.8	4.45
		45, 6	19, 41	4	17	60.8	4.02
		0, 42	6, 2	3	30	109.2	8.45
		3,2	2, 48	3	29	114.8	8.53
		21, 14	21, 37	4	17	66.6	5.03
2	mixed D∗ algorithm with Lbest PSO model	24, 24	35, 38	3	152	5823.8	26.81
		34, 38	25, 13	2	85	2854.8	13.23
		18, 21	32, 42	2	126	4544.5	20.52
		33, 18	25, 35	1	93	2909.6	17.27
		42, 3	0, 37	0	58	2021.6	8.40
	Improved approach	24, 24	35, 38	3	29	1023	10.35
		34, 38	25, 13	2	26	756.2	9.70
		18, 21	32, 42	2	25	735.2	9.56
		33, 18	25, 35	1	27	865.4	10.95
		42, 3	0, 37	0	31	1075	8.87
	hybrid AD with GSO approach	24, 24	35, 38	3	18	64	4.87
		34, 38	25, 13	2	17	25.4	4.00
		18, 21	32, 42	2	16	53	3.77
		33, 18	25, 35	1	19	60.4	5.07
		42, 3	0, 37	0	19	68.8	4.57
3	mixed D∗ algorithm with Lbest PSO model	28, 7	26, 26	2	94	2545.6	16.90
		14, 48	33, 16	1	88	2468.2	26.71
		43, 47	23, 15	1	66	1533.4	11.90
		8, 0	48, 46	0	61	2323.4	10.28
		48, 24	1,24	0	71	2879.6	11.17
	Improved approach	28, 7	26, 26	2	23	576.4	9.46
		14, 48	33, 16	1	28	871.6	10.70
		43, 47	23, 15	1	24	697.2	9.98
		8, 0	48, 46	0	31	1184.6	12.9
		48, 24	1,24	0	38	1566.2	13.12
	hybrid AD with GSO approach	28, 7	26, 26	2	15	47.6	4.69
		14, 48	33, 16	1	18	59.8	5.92
		43, 47	23, 15	1	15	54.4	4.7
		8, 0	48, 46	0	21	72	5.92
		48, 24	1,24	0	24	80.2	6.55



4	mixed D* algorithm with Lbest PSO model	3, 6	47, 14	2	106	3204.8	27.95
		25, 22	47, 14	1	65	1455	10.16
		41, 1	2, 3	1	57	1885.6	11.17
		5,49	18,0	0	54	1798	8.48
		27, 1	16, 48	0	46	1194	7.21
	Improved approach	3, 6	47, 14	2	30	1099.8	10.46
		25, 22	47, 14	1	20	455.6	6.20
		41, 1	2, 3	1	24	726.8	9.73
		5,49	18,0	0	28	945.6	11.04
		27, 1	16, 48	0	23	621.4	9.45
	hybrid AD with GSO approach	3, 6	47, 14	2	21	72.8	4.69
		25, 22	47, 14	1	13	45	2.61
		41, 1	2, 3	1	17	56.4	3.91
		5,49	18,0	0	19	64.6	4.74
		27, 1	16, 48	0	16	51.6	3.99
5	mixed D* algorithm with Lbest PSO model	22, 13	27, 17	1	84	3321.4	11.56
		33, 36	22, 12	1	73	2250.2	11.53
		11, 42	31, 8	1	75	2446.8	17.10
		1, 48	49, 1	0	70	3058.4	11.56
		26, 4	23, 48	0	66	8484	10.87
	Improved approach	22, 13	27, 17	1	33	1282.2	13.28
		33, 36	22, 12	1	24	665.2	7.56
		11, 42	31, 8	1	28	877.6	9.68
		1, 48	49, 1	0	36	1557.8	13.78
		26, 4	23, 48	0	35	1363.8	13.86
	hybrid AD with GSO approach	22, 13	27, 17	1	21	71.4	5.95
		33, 36	22, 12	1	16	53.8	3.15
		11, 42	31, 8	1	17	59.2	4.52
		1, 48	49, 1	0	25	82.8	6.31
		26, 4	23, 48	0	23	76.2	6.10
6	mixed D* algorithm with Lbest PSO model	48, 49	1, 49	0	46	1131	6.23
		0, 49	0, 3	0	45	1082.2	5.25
		0, 0	49, 49	0	63	2450.4	9.48
		0, 49	49, 0	0	63	2446.8	9.59
		39, 2	15, 37	0	53	1762.6	8.64
	Improved approach	48, 49	1, 49	0	23	578	6.98
		0, 49	0, 3	0	23	554	6.46
		0, 0	49, 49	0	33	1295	12.10
		0, 49	49, 0	0	32	1349.6	12.45
		39, 2	15, 37	0	27	934	10.93
	hybrid AD with GSO approach	48, 49	1, 49	0	16	46	3.45
		0, 49	0, 3	0	16	45	3.47
		0, 0	49, 49	0	23	79	5.41
		0, 49	49, 0	0	22	79.6	5.42
		39, 2	15, 37	0	18	65.4	4.68

The comparison results in table (4.11) illustrate that the average of occupy time is 4.81 second for the experiments of the proposed approach. While, for the same experiments the average of occupy time for the hybrid D∗ algorithm with Lbest PSO and the hybrid improved D* with Lbest PSO are 23.10 second, 10.616 second; respectively.

In addition, the average of path cost of the proposed algorithm is 19.27, which is the best value compared with the averages that are obtained by applying both the hybrid D∗ algorithm with Lbest PSO and the hybrid improved D* with Lbest PSO algorithms which have been 84.17 and 29.47, respectively.

On the other hand, Using the first improvement approach for this research study is significantly reduced the number of iterations that are required to find the optimal path for the robot, as the average number of iterations required became (19.33) iterations, while for the same experiments the average of occupy time for the hybrid mixed D∗ algorithm with Lbest PSO and the hybrid improved D* with Lbest PSO are 84.16, 29.46 iterations respectively.

The point to be noted here is that the size of individual neighbor swarm for each current node depends on the environment nature. The number of nodes that need to be selected to identify the next current node is equal to or more than (nt), which that mean it is flexible no fixed at each time. Therefore, the capacity storage requirement will be decreased, and the time occupy will be reduced as well as the number of iterations for finding the optimal path. Consequently, the proposed approach affords an improvement in the terms of robot path planning.

On the other words, the first improvement proposed approach of this study is reduced the number of iterations that are required to find the optimal path for the robot, by about 62%, compared to the hybrid mixed D∗ algorithm with Lbest PSO, and 20.83% compared to the hybrid improved D* with Lbest PSO. As for the cost of the path, it was reduced by 62% compared to the hybrid mixed D∗ algorithm with Lbest PSO, while it was reduced by 20% compared to the hybrid improved D* with Lbest PSO. finally, in terms of time occupy, it was reduced by 65% compared to the hybrid mixed D∗ algorithm with Lbest PSO and 37% compared to the other approach.

The next part of this study, and in order to clarify the effectiveness of the proposed approach for this study in finding the optimal path compared with other methods, we will review some diagrams that clearly illustrate this based on the results obtained using the Mean function.

The following three figures illustrate the comparative results among the three approaches based on the number of iterations, path cost and the occupy time respectively. Where the results obtained from applying the first approach were represented using bars in blue, while the results of the second approach were represented in green bars, and the results of the proposed approach were represented in black bars. Also, the values of Mean of each approach were added to figures. These represented figures had been appeared that the proposed approach has the best results in all these comparisons process.

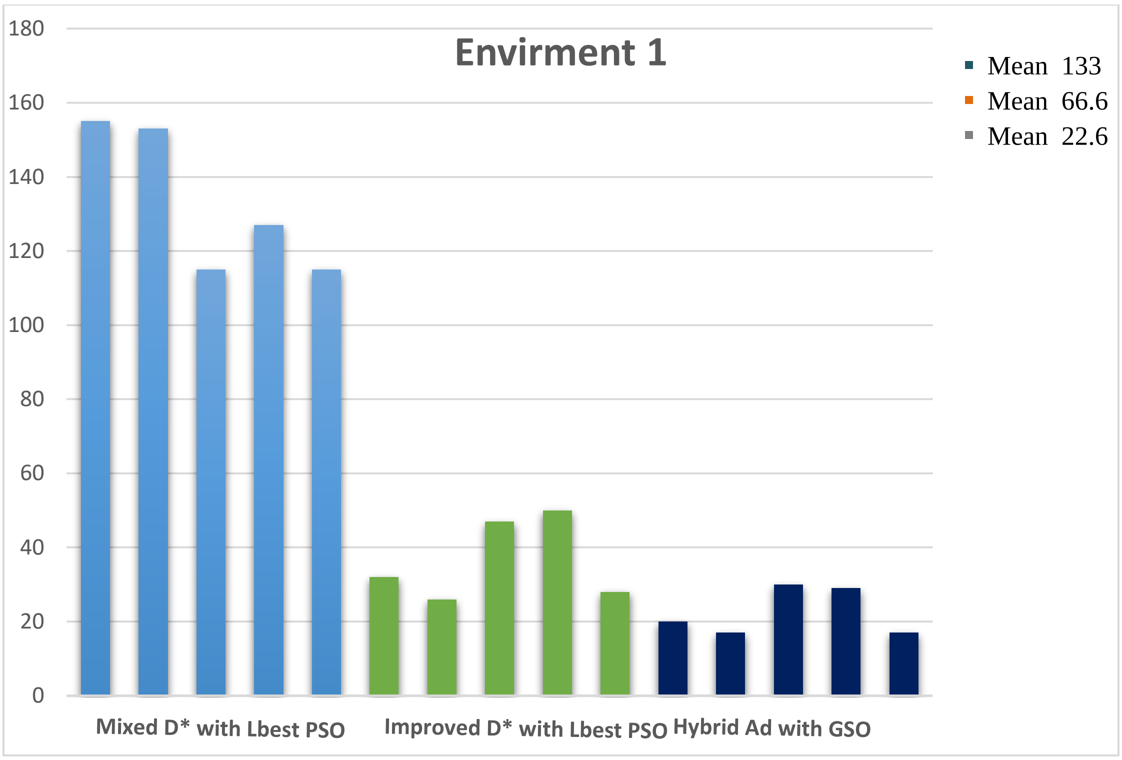


Figure (4.5) Comparison the no. of iterations to variant no. of gates in environment 1 for three improvement approaches.

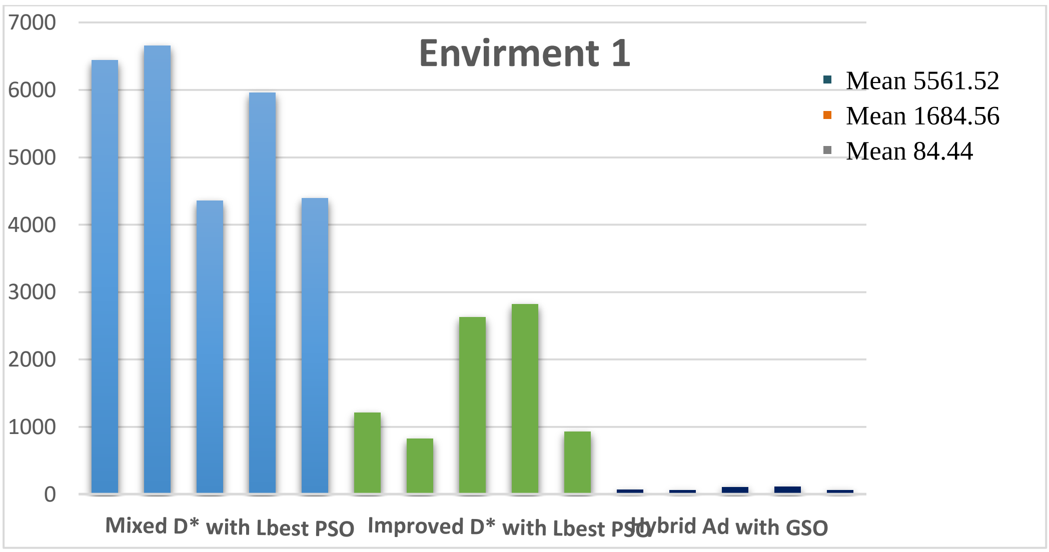


Figure (4.6) Comparison the Path cost to variant no. of gates in environment 1 for three improvement approaches.

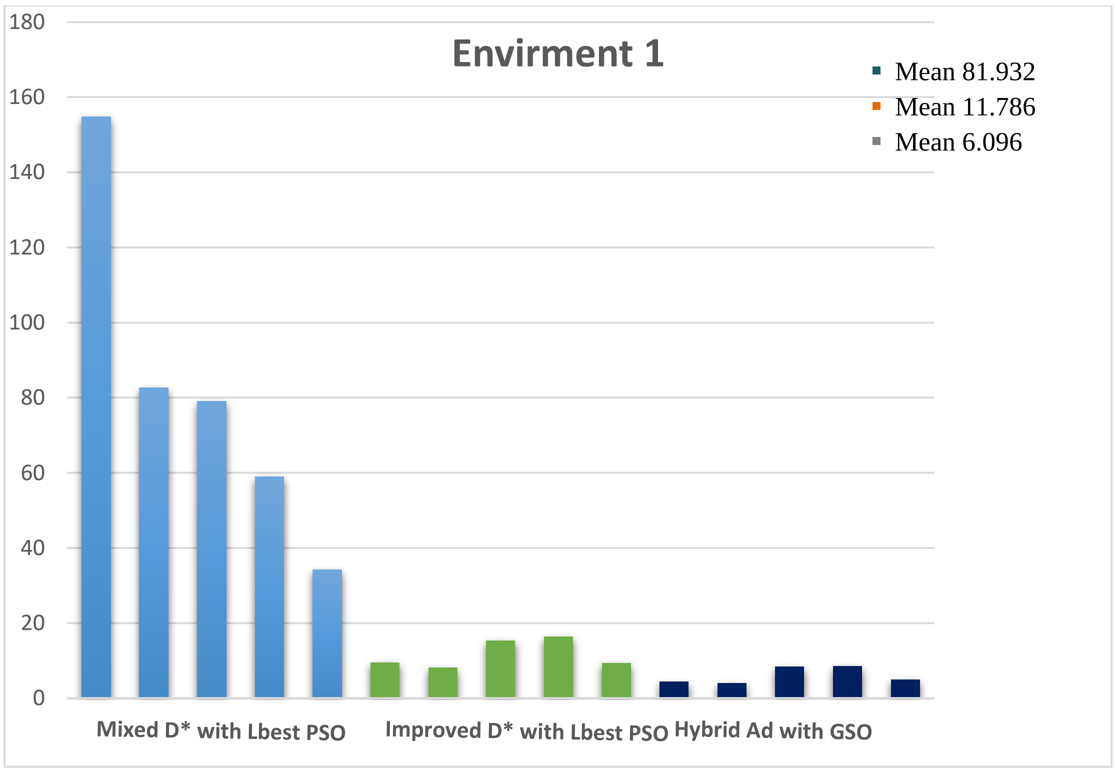


Figure (4.7) Comparison the occupy time to variant no. of gates in environment 1 for three improvement approaches.

4.5 The second improvement proposed approach.

In this Improvement approach, the work in the first improvement approach of this research study is an extended in order to adaptive the mobile robot velocity based on two factors; the number of obstacles neighbor in the closest two levels at each path node and the value of rotation angle at it, which as mentioned in the previous chapter. This extension includes many steps that must be added to the robot's movement stage exclusively to achieve the desired goal, which was mentioned in the algorithm (3.13) of the robot's movement with adapting its speed.

4.5.1 Guiding example for the second proposed approach

The following part produces an example that show how the proposed improvement approaches operate. This example is implemented to present results in details in order to show how the velocity of the mobile robot is adaptive based on the obstacles neighbour at each path node and the value of rotation angle at this node.

Static obstacles in this environment shown in black color. While, the selected start state(S) is marked by the red color at the location (0,9), goal state(G) is marked by the gold at (7,0) and finally the dynamic obstacle shows in Deep pink color. Firstly, the robot start moving from start state (0,9) with velocity value equal 0. The cumulative speed of the robot at the next current node(ve), will be calculated by calling the Algorithm (3.6).

secondly, the velocity of the robot will be computed and compared with (ve) to select the minimal one by calling Algorithm (3.9). This computation is accomplished based on the number of obstacles that lay at the closest two levels of the current node and in the direction of robot motion. On the other hand, Algorithm (3.8) is used to select the minimum value of (ve) and the robot velocity computed based on the value of rotation angle at each path node starting from the second path node**.**

According to the guiding example the values of cumulative robot velocity are ranged between 0 to 80 velocity unit, while the number of surrounding obstacle is ranged between 3 to 5 obstacles based on the nature of environment that the robot path passes through it.

On the other hand, Obstacle velocity is ranged between 53 to 62. The robot velocity based on the value of rotation angle at the current node is changed between 60 to 80. The value of adaptive robot velocity at each path node is ranged between 0 to 62 velocity unit. Finally, the average of adaptive robot velocity became 51.33. These results are shown in the Table (4.12).

Table (4.12) The results of using the second proposed approach to adapt the robot velocity for the guiding example.

Iter _no	Node no.	cumulative robot velocity	Obstacle No. at lev. 1	Obstacle No. at lev. 2	Obstacle velocity	Rotation angle	Angle velocity	Adaptive velocity	Average velocity
	(0,9)	0	-		-	-	-	0	51.33
1	(0,8)	20	3	2	56	180	80	20	
	(0,7)	40	4	1	53	180	80	40	
2	(0,6)	60	4	1	53	135	60	53	
	(1,5)	80	3	3	53	180	80	53	
	(2,4)	80	3	1	59	135	60	59	
3	(2,3)	80	3	1	59	180	80	59	
	(2,2)	80	3	0	62	180	80	62	
	(2,1)	80	2	3	59	135	60	59	
	(3,0)	80	4	1	53	135	60	53	
	(4,0)	80	4	1	53	180	80	50	
	(5,0)	70	3	1	61	180	80	40	
	(6,0)	0						0	

4.5.2 The effect of dynamic obstacle appearance on the robot velocity.

The appearance of dynamic obstacle in the neighborhood of some path nodes that is started from (0,9) and ending at (7,0). Here, the effect of the dynamic obstacles will be clearly visible at the nodes (0, 6), (2, 4), where the number of obstacles within the second level of the node (0,6) will be increased by 1 to become 2 instead of 1, while the node

(2,4) the number of obstacles within its first level will be increased by 1 to become 4 instead of 3. This increase in the number of obstacles at these nodes led to a decrease in the speed of the robot to adapt it to the increase in the number of adjacent obstacles to become 50 instead of 53 at node (0,6) and 53 instead of 59 at node (2,4). On the other hand, the average of adaptive speed of the robot will be reduced from 51.33 to 50.083. Table (4.13) show the results of application the second improvement approach with dynamic obstacle appearance.

Table (4.13) The results of applicating the second improvement approach with dynamic obstacle appearance.

Node no.	Robot ve	Obstacle No. at lev. 1	Obstacle No. at lev. 2	Obstacle velocity	Rotation angle	Angle velocity	Adaptive Velocity	Average velocity
(0,9)	0	-		-	-	-	0	50.083
(0,8)	20	3	2	56	180	80	20	
(0,7)	20	4	1	53	180	80	40	
(0,6)	40	4	2	50	135	60	50	
(1,5)	50	3	3	53	180	80	53	
(2,4)	53	4	1	53	135	60	59	
(2,3)	59	3	1	59	180	80	59	
(2,2)	59	3	0	62	180	80	62	
(2,1)	62	2	3	59	135	60	59	
(3,0)	59	4	1	53	135	60	53	
(4,0)	53	4	1	53	180	80	50	
(5,0)	50	3	1	61	180	80	40	
(6,0)	0						0	

4.5.3 Simulation Results Evaluation of the second proposed improvement approach.

In this part, the effectiveness of the proposed approach will be gaged. By applying the proposed approach on four different obtained path in three 50x50 of different environments. Some simulations will be introduced to achieve that in different situations by which the feasibility of the new approach is validated. The results of this simulation will be illustrated thus proof the effectiveness of our approach as follows:

1. Appling a proposed approach on the obtained path1 that started with (28, 23) as start state and (38, 26) as goal state which it lies in the environment 1.

This path is located within an area that is very dense with obstacles. Therefore, the number of neighborhood obstacles that surrounding the path nodes are high ranged between 2 to 8, This restricts the robot speed that depends on the number of surrounding obstacles to reach its highest value at 68 velocity unit. the robot velocity average on this path is low 45.83 velocity unit. Table (4.14) shows this results.

Table (4.14) shows the results of applicating a proposed approach on path1**.**

Node no.	Robot ve	Obstacle no. at lev. 1	Obstacle no. at lev. 2	Obstacle velocity	Rotation angle	Angle velocity	Adaptive velocity	Average velocity
(28,23)	**0**	**-**		**-**	**-**	**-**	**0**	**48.58**
(29,23)	**20**	**6**	**2**	**38**	**180**	**80**	**20**	
(30,23)	**40**	**6**	**2**	**38**	**180**	**80**	**38**	
(31,23)	**38**	**6**	**1**	**41**	**180**	**80**	**41**	
(32,23)	**41**	**5**	**3**	**41**	**135**	**60**	**41**	
(33,22)	**41**	**4**	**0**	**56**	**180**	**80**	**56**	
(34,22)	**56**	**2**	**4**	**56**	**135**	**60**	**56**	
(35,23)	**56**	**5**	**0**	**50**	**135**	**60**	**50**	
(36,23)	**50**	**3**	**1**	**59**	**180**	**80**	**59**	
(37,23)	**59**	**3**	**0**	**62**	**180**	**80**	**62**	
(38,23)	**62**	**1**	**1**	**71**	**135**	**60**	**60**	
(39,24)	**60**	**2**	**0**	**68**	**135**	**60**	**60**	

(39,25)	**60**	**2**	**5**	**53**	**135**	**60**	**40**	
(38,26)	**0**						**0**	

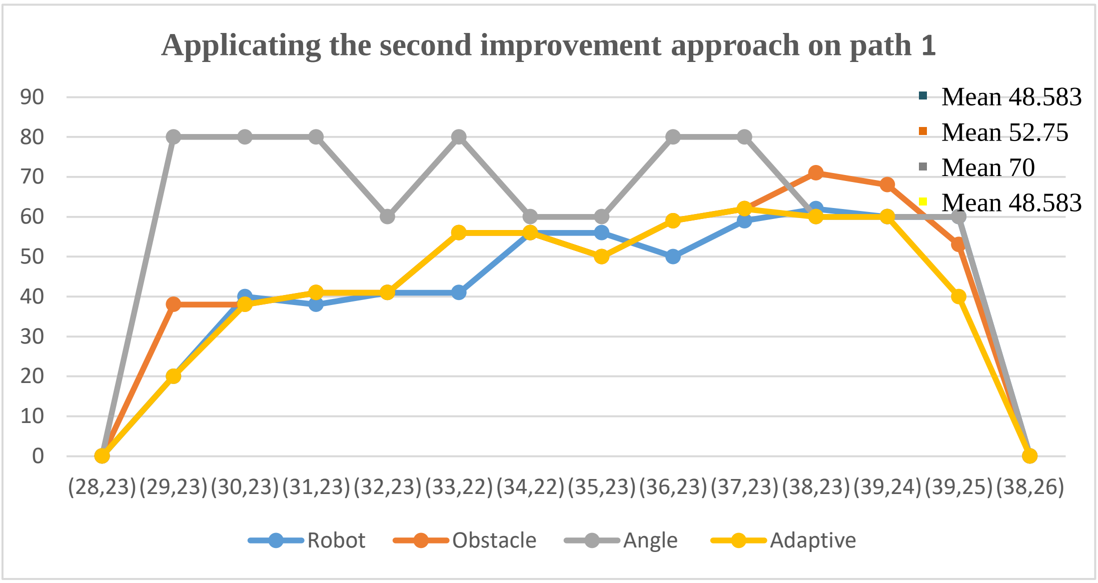


Figure (4.8) Applicating the second improvement approach on path 1

2- Now, we apply the new approach on path 2 that is located in area with less obstacles' intensity. This path starts with (2, 4) and ends with (16,0), also it lies in the same environment.

The results show that the velocity values which is based on the number of obstacles that are surrounding the current node their values range from 44 to 68 and the average of adaptive velocity is ranged between 20 to 60 velocity unit. The average of adaptive velocity is 52.12 velocity. Table (4.15) shows the results of applicating the second proposed improvement approach on path 2.

Table (4.15) results of applicating the second proposed improvement approach on path 2.

Node no.	Robot ve.	Obstacle no. at lev. 1	Obstacle no. at lev. 2	Obstacle velocity	Rotation angle	Angle velocity	Adaptive velocity	Averag e velocity
(2,4)	0	-		-	-	-	0	55.08
(3,5)	28	3	2	56	180	80	28	
(4,6)	28	3	0	62	135	60	56	
(5,6)	56	2	2	62	135	60	60	
(6,5)	60	3	2	56	135	60	56	
(7,5)	56	3	1	59	180	80	59	
(8,5)	59	4	2	50	180	80	50	
(9,5)	50	5	1	47	180	80	47	
(10,5)	47	4	1	53	180	80	53	
(11,5)	53	4	0	56	180	80	56	
(12,5)	56	2	2	62	135	60	60	
(13,4)	60	3	1	59	135	60	59	
(14,4)	59	2	1	65	180	80	65	
(15,4)	65	2	0	68	180	80	68	
(16,4)	68	2	1	65	135	60	60	
(17,3)	60	2	0	68	135	60	60	
(17,2)	60	2	3	59	135	60	59	
(16,1)	59	3	2	56	135	60	40	
(16,0)	40						0	

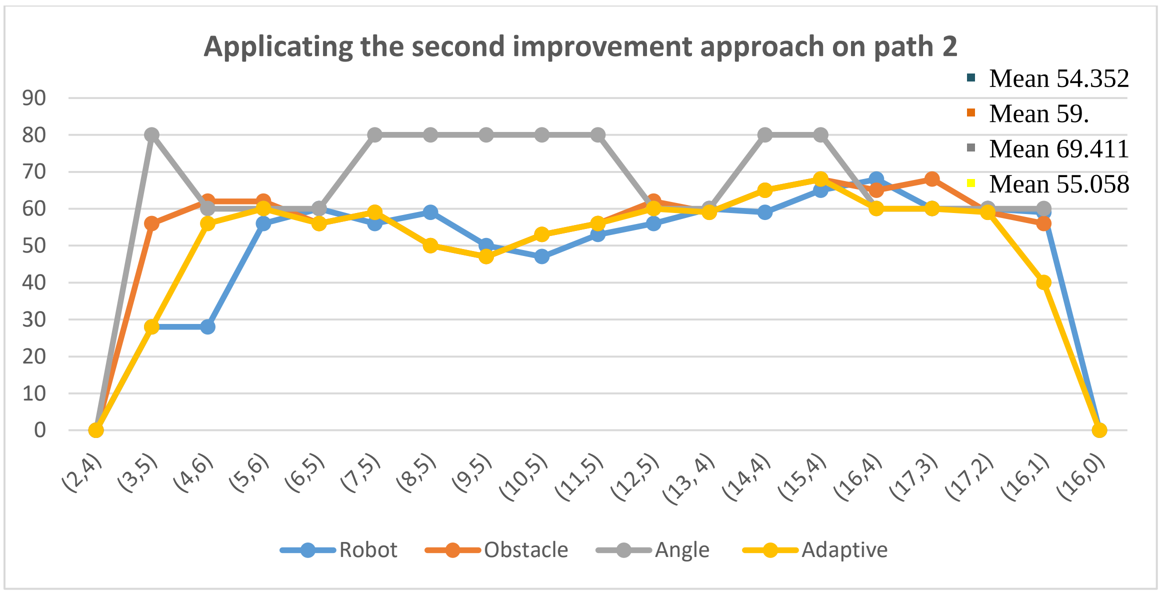


Figure (4.9) Applicating the second improvement approach on path 2

3-When the second proposed improvement approach was applied to the third path, which is located in the second environment within a semi-obstacle-free area.

Through the results that appeared in the table (4.16) below, we note that the permissible velocity range based on the number of obstacles surrounding the current node and the motion direction increased to range between 62 to 80 velocity units. The velocity based on the values of the angles at each node, they are between 60 and 80 velocity units. This leaded to an increase in the average of adaptive speed to become 65.09 units of speed and thus be more realistic in proportion to the nature of the surrounding environment.

Table (4.16) results of application the second proposed improvement approach on path 3.

Node no.	Robot ve	Obstacle no. at lev. 1	Obstacle no. at lev. 2	Obstacle velocity	Rotation angle	Angle velocity	Adaptive velocity	Average velocity
(34,15)	0	-		-	-	-	0	65.36
(35,16)	0	0	0	80	180	80	28	
(36,17)	28	0	0	80	180	60	56	
(37,18)	56	0	0	80	180	80	80	
(38,19)	80	0	1	77	180	80	77	
(39,20)	77	1	1	71	180	80	71	
(39,21)	71	2	0	68	180	80	68	
(39,22)	68	2	1	65	180	80	65	
(40,23)	65	2	0	68	180	80	68	
(41,23)	68	2	0	68	135	60	60	
(42,24)	60	0	0	80	180	80	80	
(43,25)	80	0	0	80	180	80	80	
(44,26)	80	0	0	80	180	80	80	
(45,27)	80	0	0	80	180	80	80	
(46,28)	68	0	1	77	135	60	60	
(46,29)	60	1	2	71	180	80	71	
(46,30)	71	2	0	68	180	80	68	
(46,31)	68	2	1	65	135	60	60	
(45,32)	60	2	1	65	135	60	60	
(44,32)	60	4	0	56	135	60	56	
(44,33)	56	1	0	74	180	80	74	
(44,34)	74	0	0	80	180	80	80	
(44,35)	80	0	0	80	180	0	40	
(44,36)	40						0	

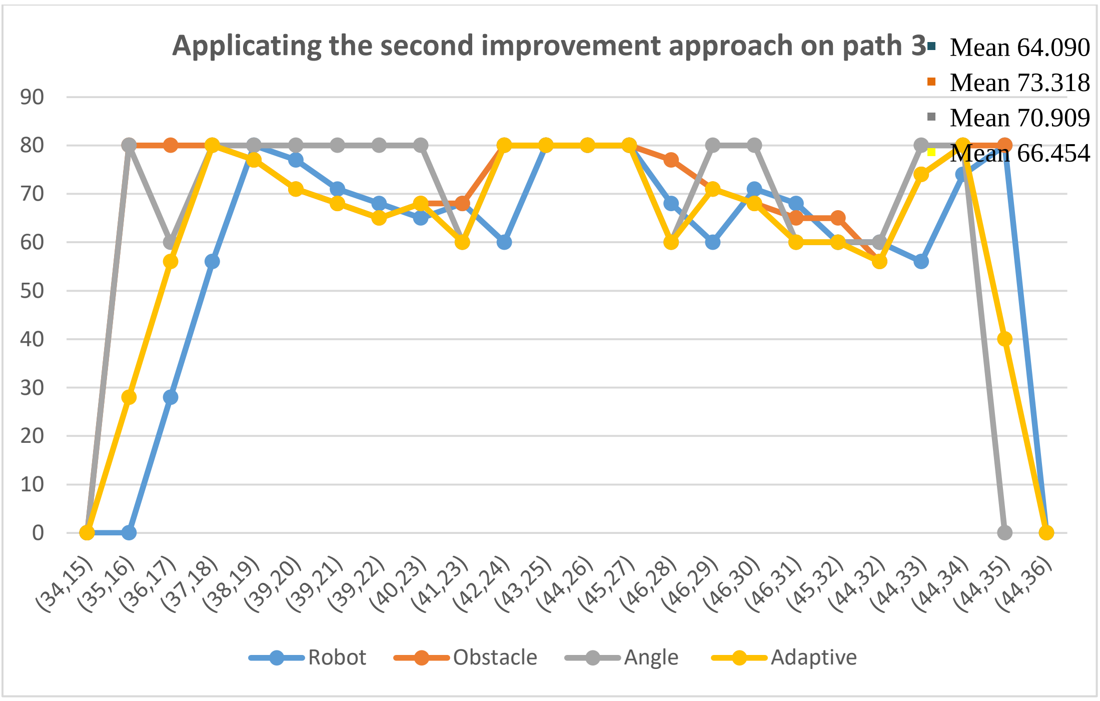


Figure (4.10) Applicating the second improvement approach on path 3

4- Finally, when the new approach is applied on the fourth path that beginning from (34, 12) until (35, 36) which lies in the third environment with size 50x50. On the other hand, this path is located in free-obstacles area.

Beyond applying the second research approach on the fourth path, the obtained results from this application for robot velocity's adaptation to meet its environment's nature have been introduced by table (4.17) neatly.

Table (4.17) results of applicating the second proposed improvement approach on path 4.

Node no.	Robot ve	Obstacle no. at lev. 1	Obstacle no. at lev. 1	Obstacle velocity	Rotation angle	Angle velocity	Adaptive velocity	Average velocity
(34,12)	0	-		-	-	-	0	74.08
(34,13)	28	0	0	80	135	60	28	
(35,14)	28	0	0	80	180	80	48	
(35,15)	48	0	0	80	180	80	64	
(35,16)	64	0	0	80	180	80	80	
(35,17)	80	0	0	80	180	80	80	
(35,18)	80	0	0	80	180	80	80	
(35,19)	80	0	0	80	180	80	80	
(35,20)	80	0	0	80	180	80	80	
(35,21)	80	0	0	80	180	80	80	
(35,22)	80	0	0	80	180	80	80	
(35,23)	80	0	0	80	180	80	80	
(35,24)	80	0	0	80	180	80	80	
(35,25)	80	0	0	80	180	80	80	
(35,26)	80	0	0	80	180	80	80	
(35,27)	80	0	0	80	180	80	80	
(35,28)	80	0	0	80	180	80	80	
(35,29)	80	0	0	80	180	80	80	
(35,30)	80	0	0	80	180	80	80	
(35,31)	80	0	0	80	180	80	80	
(35,32)	80	0	0	80	180	80	80	
(35,33)	80	0	0	80	180	80	80	
(35,34)	80	0	0	80	180	80	40	
(35,35)	40	0	0	80	180	80	40	
(35,36)	40						0	

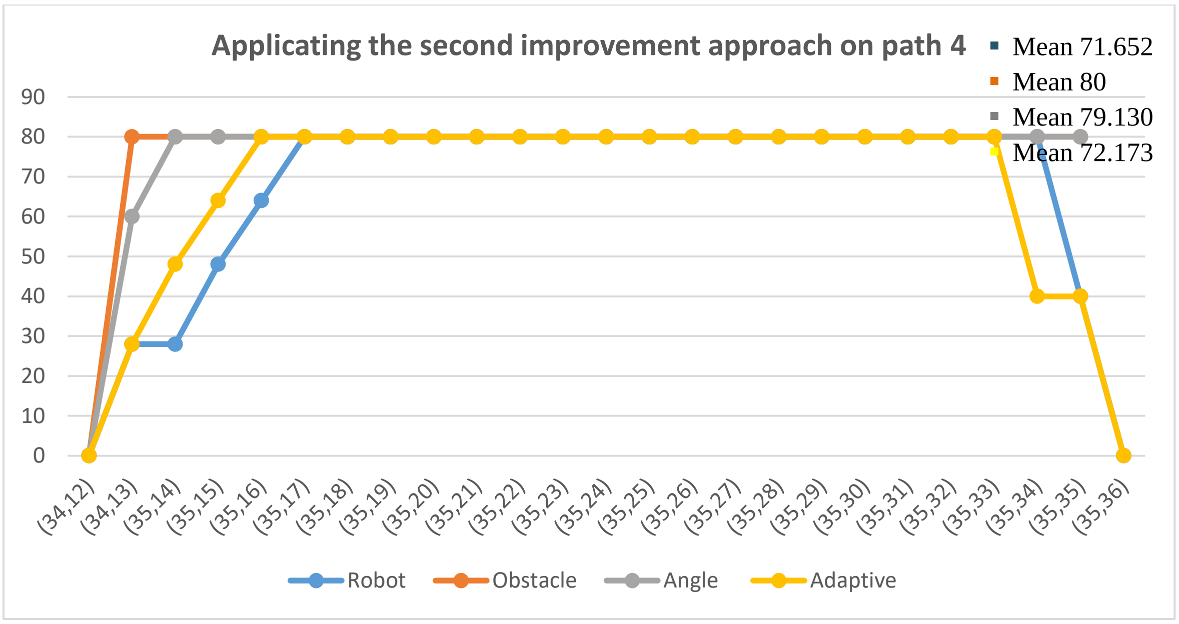


Figure (4.11) Applicating the second improvement approach on path 4

By taking a closer look at the results in the above tables, we can say that the second proposed approach achieved good results for optimizing the robot's velocity by a percentage that sometimes exceeds 300%. table (4.18) above shows the results obtained by applying the second improved approach of this dissertation on the four paths within its work environment that differ in terms of the intensity of the surrounding obstacles and the value of the rotation angle at each node of the path. The minimum velocity that the robot can travel is 20 units of speed which is one of the characteristics of the robot that has been assumed previously.

Table (4-18) The comparison results

Approach Name	Path	Lowest Velocity	Adaptive Velocity	Percentage
Robot's Adaptive Velocity for planned path by hybrid adaptive dimensionality with GSO	**1**	**20**	**45.3**	**226.5%**
	2	**20**	**52.12**	**260.6%**
	3	**20**	**66**	**330%**
	4	**20**	**76.2**	**381.5%**
				300%
Adaptive Robot speed control by Considering Map and Localization Uncertainty		**0.3**	**0.9**	**300%**

By taking a closer look at the results in the above tables, we can say that the second proposed approach achieved good results for optimizing the robot's velocity by a percentage that sometimes exceeds 300%. table (4.18) above shows the results obtained by applying the second improved approach of this dissertation on the four paths within its work environment that differ in terms of the intensity of the surrounding obstacles and the value of the rotation angle at each node of the path. The minimum velocity that the robot can travel is 20 units of speed which is one of the characteristics of the robot that has been assumed previously.

The first path passes through a very dense area with obstacles, so the speed of the robot has been adapted to 45.3 velocity units, an increase of 25.3 velocity units compare w the minimum speed that the robot can move towards the goal. While in the second path, the robot passed in a less dense obstacles environment and rotation angles with different values than in the first path, which led to an increase in its average speed, reaching 52.12 speed units.

A remarkable decrease in the number of obstacles that are surrounding the third path nodes, which are located at the two closest levels to its nodes, which led to an increase in the robot's speed while traveling between the path nodes reaching the goal to become about 66 speed units.

As for the track extending between the two knots (12, 34), (36, 35), which passed through an area almost free of obstacles, so that its speed was close to its highest assumed speed (80 speed units), which reached 76.2 speed units.

Finally, by calculating the average robot speed, which was obtained through the four aforementioned simulations, it was found equal to 60 speed units, which is 3 times the minimum speed that the robot can move by it. Therefore, the percentage of its velocity increase amounted to 300%, which is equal to the percentage obtained using approach that proposed in [6] and this indicates the effectiveness of the proposed approach for this research.

Chapter Five

CONCLUSIONS AND FUTURE WORKS

5.1 Introduction

In this dissertation, contributions related to the controlling of robot velocity and finding optimal path depending on glowworm optimization are presented. This chapter concentrates on the main conclusions derived from the work and the future directions presented in sections 5.2 and 5.3.

5.2 Conclusion

1. Using an appropriate data structure significantly reduces both a storage space and the time that are required to perform the calculations to find the robot path or to adaptive the robot velocity. This study also develops method of data structure to the state space and the values of dimensions for each represented unit in the space. The efficiency of this data structure will influence the proposed approaches to reduce the size of the required storage space and the time to find an optimal path.

2. Taking the factors that affect the robot velocity during its movement toward its goal within its work environment ensures, in a remarkable way, a successful adaptation process to its velocity, which leads to increase in its speed and move safely without colliding or falling.
3. Using an appropriate data structure significantly reduces both a storage space and the time that are required to perform the calculations to find the robot path or to adaptive the robot velocity. This study also develops method of data structure to the state space and the values of dimensions for each represented unit in the space. The efficiency of this data structure will influence the proposed

approaches to reduce the size of the required storage space and the time to find an optimal path.

4. In this dissertation, simulation experiments have been applied to many dynamic environments that differ in terms of size, number of gates and density of obstacles that differ in terms of size, shape, and type(static and dynamic). All the simulation experiments that were conducted to find the optimal path proved that the first proposed approach for this dissertation is the efficient, compared with the other two approaches which as follows:

A. Using the first improvement approach for this research study is significantly reduced the number of iterations that are required to find the optimal path for the robot, as the average number of iterations required became (19.33) iterations, while for the same experiments the average of execution time for the hybrid mixed D∗ algorithm with Lbest PSO and the hybrid improved D* with Lbest PSO are 84.16, 29.46 iterations respectively.

B.Additionally, the average of path cost of the proposed algorithm is 19.27, which is the best value compared with the averages that are obtained by applying both the hybrid D∗ algorithm with Lbest PSO and the hybrid improved D* with Lbest PSO algorithms which have been 84.17 and 29.47, respectively.

C. Finally, the comparison results illustrate that the average of execution time is 4.81 seconds for the experiments of the proposed approach. While, for the same experiments the average of execution time for the hybrid D∗ algorithm with Lbest PSO and the hybrid improved D* with Lbest PSO are 23.10 second and 10.616 second respectively.

Also, the second improvement approach of this dissertation is able to adapt the robot velocity to become as 48.58 velocity unit when the robot path pass through when the robot path passes through a very obstacles-dense environment. On the other hand, when the robot moves through semi obstacle-free area, its velocity is increased to become about 60.22. Additionally, when the robot moves in free-obstacles area, its velocity is increased so much to nearly from its maximum value (80 velocity unit) to become 74.08 velocity. Also, the results show that the second proposed improvement approach of this dissertation adapt the robot velocity to increase it by 300% in comparing with the low robot velocity (20 velocity unit).

5.2 Suggested Future Works

There are many suggestions that can be achieved in the future which as follows:

1. Increasing the number of dimensions by adding other useful dimensions that allow to take smarter decisions to find the optimal path, including the nature of the environment, determining the densest obstacles' areas, and trying to choose paths within the least dense areas as much as possible to improve the speed of the robot, but not at the expense of an increase in the cost of the path and according to weights and measurements to be determined.

2. An additional mechanism can be used to adapt the robot’s speed by moving it away from obstacles, at least one node, during its movement to avoid gradient in its speed as much as possible.

3. A good future direction is implementing another GSO algorithms and studying their features with robot path planning problem to determine the most efficient algorithm.

REFERENCES

[1] G. Vasilev, "Planning With Adaptive Dimensionality". Publicly Accessible Penn Dissertations. 1739, https://repository.upenn.edu /edissertations /1739/, 2016.

[2] H. Fatin et al., “Multi-Objective Path Planning of an Autonomous Mobile Robot using Hybrid PSO-MFB Optimization Algorithm”, April 2020Applied Soft Computing 89:106076, 2019.

[3] K. Karthik et al.,“A Survey of Path Planning Algorithms for Mobile Robots”, Vehicles 2021, 3, 448–468. https://doi.org/10.3390/vehicles 3030027, 2021.

[4] Alessandro et al., “Path Planning and Trajectory Planning Algorithms: a General Overview”, Vidoni Motion and operation Planning of Robotic Systems, chapter1, pp.3-27, 2015, Springer International Publishing, 2015.

[5] C. Woojin et al., “Safe Navigation of a Mobile Robot Considering Visibility of Environment”, IEEE Transactions on Industrial Electronics, Vol. 56, NO. 10, October 2009.

[6] Z. Han-ye, L. Wei-ming and C. Ai-xia, “Path Planning for the Mobile Robot: A Review”, Symmetry 2018, 10, 450; doi:10.3390/sym10100450, 2018.

[7] M. Contreras-Cruz, V. Ayala-Ramirez and U. Hernandez-Belmonte, Mobile robot path planning using artificial bee colony and evolutionary programming. Appl. Soft Comput. 30, 319–328. [CrossRef], 2015.

[8] K. Karthik., S. Nitin, D. Chinmay and E. Joshua, “A Survey of Path Planning Algorithms for Mobile Robots”, Vehicles 2021, 3, 448–468. https://doi.org/10.3390/ vehicles3030027, 2021.

[9] N. Siti, T. Bambang, “Intelligent Robotics Navigation System: Problems, Methods, and Algorithm”, International Journal of Electrical

and Computer Engineering (IJECE), Vol. 7, No. 6, pp. 3711~3726, December 2017.

[10] F. Amalia and E. Panos, "Predictive Control of Robot Velocity to Avoid Obstacles in Dynamic Environments", Proceedings of the IEEE/RSJ Intl. Conference on Intelligent Robots and Systems Las Vegas, Nevada • 2003.

[11] N. Yoshiro, M. Jun, and S. Yoshiaki, "Adaptive Robot Speed Control by Considering Map and Localization Uncertainty", Proceedings of the 8th Int. Conf. on Intelligent Autonomous Systems, pp. 873-880, Amsterdam, The Netherlands, March 2014.

[12] L. He, and S. Huang, " Improved Glowworm Swarm Optimization Algorithm for Multilevel Color Image Thresholding Problem" Hindawi Publishing Corporation Mathematical Problems in Engineering Volume 2016, Article ID 3196958, 24 pages http://dx.doi.org/10.1155/2016/3196958, 2016.

[13] K. Gochev, "Planning With Adaptive Dimensionality". Publicly Accessible Penn Dissertations. 1739. http://repository.upenn.edu/ edissertations /1739, 2016.

[14] A. Hasan, "Robot Path Planning based on Improved MAX–MIN Ant Colony Optimization Algorithm in Dynamic Environment", Research Journal of Applied Science 11(10): 1060-1068, 2016.

[15] A. Sadiq, and A. Hasan, "Robot Path Planning Based on PSO and D∗ Algorithms in Dynamic Environment". International Conference on Current Research in Computer Science and Information Technology (ICCIT), pp.145–150, 2017.

[16] S. Girija, and J. Ashok, "Fast Hybrid PSO-APF Algorithm for Path Planning in Obstacle Rich Environment", IFAC PapersOnLine 52-29 (2019) 25–30, 2019.

[17] M. Marcia and F. Manuel,"A Survey on Path Planning Algorithms for Mobile Robots", 978-1-7281-3558-8/19/$31.00 ©2019 IEEE, 2019.

[18] Z. Han-ye , L. Wei-ming and C. Ai-xia, "Path Planning for the Mobile Robot: A Review", Symmetry,10,450; doi:10.3390/sym10100450, 2018.

[19] K. Pierrick, L. Simon, "Managing environment models in multi-robot teams", International Conference on Intelligent Robots and Systems (IROS), Daejeon, South Korea. pp.5722 - 5728, 2017.

[20] M. Habib, and H. Asama, "Efficient method to generate collision free paths for an autonomous mobile robot based on new free space structuring approach", In Proceedings of the IEEE/RSJ International Workshop on Intelligent Robots and Systems (IROS'91), Osaka, Japan,; pp. 563–567, 1991.

[21] N. Fadzlina, "Probabilistic roadmap based path planning for mobile robots", 2014.

[22] R. Almadhoun et al, "A survey on multi-robot coverage path planning for model reconstruction and mapping", SN Applied Sciences, 1:847 | https://doi.org/10.1007/s42452-019-0872-y., 2019.

[23] K. Akka, F. Khaber, "Mobile robot path planning using an improved ant colony optimization", International Journal of Advanced Robotic Systems , 2018.

[24] M. Contreras-Cruz, V. Ayala-Ramirez and U. Hernandez-Belmonte, , "Mobile robot path planning using artificial bee colony and evolutionary programming", Appl. Soft Comput, vol .30, pp.319–328 , 2015.

[25] A. Koubaa et al, "Introduction to Mobile Robot Path Planning", Springer International Publishing AG, part of Springer Nature 2018

[26] S. Wu , Y. Du , and Y. Zhang "Mobile Robot Path Planning Based on a Generalized Wavefront Algorithm", Mathematical Problems in

Engineering Volume 2020, Article ID 6798798, 12 pages https://doi.org/10.1155/2020/6798798, 2020.

[27] D. Verma, and P. Saxena, "Robot navigation and target capturing using nature-inspired approaches in a dynamic environment", arXiv:1911.02268v1 [cs.AI], 2019.

[28] M. Kleinbort et al, "Probabilistic completeness of RRT for geometric and kinodynamic planning with forward propagation", arXiv:1809.07051v1 [cs.RO], 2018.

[29] P. Teleweck1, and B. Chandrasekaran "Path Planning Algorithms and Their Use in Robotic Navigation Systems", IOP Conf. Series: Journal of Physics: Conf. Series 1207, 2019.

[30] V. Kenny, M Nathal, and S. Saldana, "Heuristic algorithms", ChE 345 Spring, 2014.

[31] A. Stentz, "The Focussed D* Algorithm for Real-Time Replanning", In Proceedings of the International Joint Conference on Artificial Intelligence, 1995.

[32] S. Koenig, and M. Likhachev. "Improved fast replanning for robot navigation in unknown terrain." In Robotics and Automation, Proceedings. ICRA'02. IEEE International Conference on, vol. 1, pp. 968-975. IEEE, 2002.

[33] M. Amira et al., "Swarm Optimization Techniques", https://www.researchgate.net/publication/326440873_Swarm_Optimization_Techniques_A_survey, 2018.

[34] A. Chakraborty and A. Kar, "Swarm Intelligence: A Review of Algorithms", © Springer International Publishing AG 2017 S. Patnaik et al. (eds.), Nature-Inspired Computing and Optimization, Modeling and Optimization in Science and Technologies 10, DOI 10.1007/978-3-319-50920-4_19, 2017.

[35] Y. Tamura, T Sakiyama , and I. Arizono, “Ant Colony Optimization Using Common Social Information and Self-Memory”, Hindawi Complexity Volume 2021, Article ID 6610670, 7 pages https://doi.org/10.1155/2021/6610670, 2021.

[36] S. Chia et al., “Ant Colony System Based Mobile Robot Path Planning”, Fourth International Conference on Genetic and Evolutionary Computing, 2010.

[37] D. Wang, D. Tan, and L. Liu, “Particle swarm optimization algorithm: an overview”, © Springer-Verlag Berlin Heidelberg, 2017.

[38] Y. Shi and R. Eberhart, “A Modified Particle Swarm Optimizer ” , Conference Paper, 0-7803-4869-9198 $10.0001998 EEE, 1998.

[39] S. Kiranyaz *et. Al.,* “Fractional Particle Swarm Optimization in Multidimensional Search Space”, IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics), vol. 40, pp. 298–319, 2009.

[40] X. Li et. al., “An Improved Method of Particle Swarm Optimization for Path Planning of Mobile Robot”, Journal of Control Science and Engineering Volume, Article ID 3857894, 12 pages https://doi.org/10.1155/2020/3857894,2020.

[41] H. Dewanga , P. Mohantya, S. Kundu, “A Robust Path Planning For Mobile Robot Using Smart Particle Swarm Optimization”, International Conference on Robotics and Smart Manufacturing (RoSMa), 2018.

[42] M. Shahab, M. Usman, and M. Umer, “Mobile Robot Path Planning in Static Environments using Particle Swarm Optimization” , International Journal of Computer Science and Electronics Engineering (IJCSEE) Volume 3, Issue 3, 2015.

[43] D. Verma, and P. Saxena, “Robot navigation and target capturing using nature-inspired approaches in a dynamic environment”, arXiv:1911.02268v1 [cs.AI] , 2019.

[44] T. Kalaiselvi, P. Nagaraja, Z. Abdul Basith, “A Review on Glowworm Swarm Optimization”, International Journal of Information Technology (IJIT) – Volume 3 Issue 2, 2017.

[45] N. Zainal et. al., “Glowworm Swarm Optimization (GSO) Algorithm for Optimization Problems: A State-of-the-Art Review”, Applied Mechanics and Materials Vol. 421, pp 507-511, 2013.

[46] Z. Li and X. Huang, “Glowworm Swarm Optimization and Its Application to Blind Signal Separation”, Copyright © 2016 Z. Li and X. Huang. This is an open access article distributed under the Creative Commons Attribution License, 2016.

[47] R. Eberhart and Y. Shi, “Comparing inertia weights and constriction factors in particle swarm optimization,” in Proceedings of the Congress on Evolutionary Computation (CEC ’00), pp. 84– 88, San Diego, Calif, USA, 2000.

[48] A. S. D. Dymond, A. P. Engelbrecht, and P. S. Heyns, “The sensitivity of single objective optimization algorithm control parameter values under different computational constraints,” in Proceedings of the IEEE Congress of Evolutionary Computation (CEC '11), pp. 1412–1419, New Orleans, LA, 2011.

[49] S. Smit and A. Eiben, “Comparing parameter tuning methods for evolutionary algorithms,” in Proceedings of the IEEE Congress on Evolutionary Computing, pp. 399–406, IEEE, Trondheim, Norway, 2009.

[50] R. Thangaraj, M. Pant, A. Abraham, and P. Bouvry, “Particle swarm optimization: hybridization perspectives and experimental illustrations,” Applied Mathematics and Computation, vol. 217, no. 12, pp. 5208–5226, 2011.

[51] X. Shi, Y. Liang, H. Lee, C. Lu, and L. Wang, “An improved GA and a novel PSO-GA-based hybrid algorithm,” Information Processing Letters, vol. 93, no. 5, pp. 255–261, 2005.

[52] N. Holden and A. A. Freitas, “A hybrid particle swarm/ant colony algorithm for the classification of hierarchical biological data,” in Proceedings of the IEEE Swarm Intelligence Symposium (SIS ’05), pp. 100–107, IEEE, 2005.

[53] H. Li and L. Li, “A novel hybrid particle swarm optimization algorithm combined with harmony search for high dimensional optimization problems,” in Proceedings of the International Conference on Intelligent Pervasive Computing (IPC ’07), pp. 94– 97, 2007.

[54] A. Tariq and A. Hadi, “Robot Path Planning Based on PSO and D∗ Algorithms in Dynamic Environment”. International Conference on Current Research in Computer Science and Information Technology (ICCIT), pp.145–150, 2017.

[55] A. Tariq and A. Hadi, “Robot Path Planning Based on Hybrid Improved D∗ with PSO Algorithms in Dynamic Environment”, Journal of Computational and Theoretical Nanoscience Vol. 16, 1–12, 2019.

[56] F. Qian, M. Reza Mahmoudi, and H. Parvin, "An Adaptive Particle Swarm Optimization Algorithm for Unconstrained Optimization", Volume 2020 |Article | D 2010545 https://doi.org/10.1155/2020/2010545, 2020.

[57] Q. Luo, Z. Ouyang, X. Chen, and Y. Zhou, “A multilevel threshold image segmentation algorithm based on glowworm swarm optimization” Journal of Computational Information Systems, vol. 10, no. 4, pp. 1621–1628, 2014.

[58] K. Gochev, A. Safonova and M. Likhachev, “Planning with Adaptive Dimensionality for Mobile Manipulation”. ONR grant N00014-09-1-1052, DARPA CSSG program D11AP00275 and the

Army Research Laboratory Cooperative Agreement Number W911NF-10-2-0016, 2012.

[59] A. Vemula, K. Muelling and J. Oh, “Path Planning in Dynamic Environments with Adaptive Dimensionality”. Copyright, Association for the Advancement of Artificial Intelligence (www.aaai.org). All rights reserved, 2016.

[60] K. Gochev, A. Safonova, and M. Likhachev, “Incremental planning with adaptive dimensionality”. In ICAPS, 2013.

[61] K. Gochev, "Planning With Adaptive Dimensionality". Publicly Accessible Penn Dissertations. 1739. http://repository.upenn.edu/ edissertations /1739, 2016.

[62] Y. Li, H. Li, and G. Wei, “Dimension-adaptive algorithm-based PCE for models with many model parameters”, ISSN: 0264-4401, **2019.**

[63] D. Kim and et. al. “Efficient path planning for high-DOF articulated robots with adaptive dimensionality”, IEEE International Conference on Robotics and Automation (ICRA),2015.

[64] J. Meriam and L. Kraige, “Engineering Mechanics Dynamics”, sixth edition, 2014.

[65] H. Imen, M. Imen and R. Chokri, “Path Planning with Avoiding Obstacles in Known Environment Using Free Segments and Turning Points Algorithm”, Mathematical Problems in Engineering Volume, 2018.

[66] P. Maurice et al., “Velocity-curvature patterns limit human-robot physical interaction”, IEEE Robotics and Automation Letters, Volume: 3, Issue: 1, Jan, 2018.

[67] A. Zelenak, C. Peterson, J. Thompson., and M. Pryor, “The Advantages of Velocity Control for Reactive Robot Motion”,

Proceedings of the ASME 2015 Dynamic Systems and Control Conference DSCC2015 October Columbus, Ohio, USA, 2015.

[68] A. Foka, P. Trahanias , “Predictive Control of Robot Velocity to Avoid Obstacles in Dynamic Environments”, Proceedings of the IEEE/RSJ Intl. Conference on Intelligent Robots and Systems Las Vegas, Nevada, 2003.

[69] Y. Negishi, J. Miura, and Y. Shirai, “Adaptive Robot Speed Control by Considering Map and Localization Uncertainty”, Proceedings of the 8th Int. Conf. on Intelligent Autonomous Systems, pp. 873-880, Amsterdam, The Netherlands, March , 2014.

[70] A. Zelenak, C. Peterson., J. Thompson., and M. Pryor, “The Advantages of Velocity Control for Reactive Robot Motion”, Proceedings of the ASME 2015 Dynamic Systems and Control Conference DSCC2015 October Columbus, Ohio, USA, 2015.

[71] T. Kurosaka and M. Kaneko, “Autonomous Mobile Robot Selecting Optimum Path with Safe Speed Control in Consideration of Blind Area of Vision Sensors”, Electronics and Communications in Japan, Vol. 99, No. 8, 2016.

[72] J. Xue., C Xia. and J. Zou., “A velocity control strategy for collision avoidance of autonomous agricultural vehicles”, Autonomous Robots, Volume 44, Pages 1047-1063, 2020.

جمهورية العراق

وزارة التعليم العالي والبحث العلمي

جامعة بابل - كلية تكنولوجيا المعلومات

قسم البرمجيات

تخطيط مسار الروبوت بالاعتماد على تهجين خوارزميتي تمثيل الابعاد التكيفية مع سرب الحشرة المضيئة.

اطروحة مقدمة

الى مجلس كلية تكنولوجيا المعلومات - جامعة بابل وهي جزء من متطلبات نيل درجة الدكتوراه فلسفة في تكنولوجيا المعلومات / برمجيات

من قبل

قاسم ردام محمود عذاب العبيدي

إشراف

أ. د حسين كيطان منسي علوان

أ.م. د علي هادي حسن عباس

2022 م 1443هـ

الملخص

في أيامنا هذه ، تحتل الروبوتات اهتمامًا كبيرًا في حياة الإنسان خاصة في الصناعات الحديثة التي تعمل تلقائيًا. يعتبر تخطيط مسار الروبوت من المشكال الرئيسية لحركة الروبوت الآمنة والفعالة. على الجانب الآخر ، يعد تحسين سرعة الروبوت عند انتقال الروبوت المحمول من الموضع الأولي إلى موضع الهدف عاملاً حاسمًا في مجال الروبوتات. بالإضافة إلى ذلك ، يعد التحكم في سرعة الروبوت أمرًا مهمًا في الحالات التالية. (1) عند حركة الروبوت في المنطقة الخالية الضيقة. (2) عندما يلتف الروبوت حول زوايا مساره متجهًا إلى الهدف. تم استخدام مفهوم الأبعاد التكيفية (AD) بنجاح لحل هذا النوع من المشاكل. من ناحية أخرى ، فإن خوارزميات تحسين ذكاء السرب ناجحة إلى حد كبير في إيجاد حلول لمشاكل تخطيط مسار الروبوت.

الهدف الأول لهذا البحث هو تطوير نهج لتخطيط مسار الروبوت في فضاء الحالة الخاص بالبيئة الديناميكية التي تستند إلى الحقائق السابقة. يتركز التحسين في هذا النهج المقترح في أن عدد الجيران الذين سيتم اختيارهم لتحديد عقدة المسار التالية سيكون رقمًا متكيفًا (غير ثابت لجميع العقد). لأن عدد العقد المجاورة التي سيتم تحديدها لتحديد العقدة الحالية التالية يجب أن يكون لها سمات محددة ويجب أن تكون موجودة ضمن النطاق المرئي الأقصى (MVR).

يؤدي هذا التطور إلى تقليل سعة التخزين المطلوبة ، وبالتالي تقليل الوقت الذي تستغرقه المعالجة. تؤدي الأبعاد التكيفية أيضًا إلى تقليل عدد التكرارات المطلوبة لتحديد المسار من حالة البداية إلى حالة الهدف. يتم تحقيق الهدف الثاني من هذا البحث من خلال تطوير نهج لتحسين سرعة الروبوت بناءً على العوامل التالية ، قيمة كل زاوية دوران في كل عقدة من المسار التي نتجت عن تطبيق النهج الأول وعدد العوائق المجاورة التي تحيط بهذه العقدة ضمن أقرب مستويين لها.

يتم تحديد فعالية هذين النهجين من خلال تطبيقهما على بيئات ديناميكية مختلفة للعثور على المسار الأمثل للروبوت ومن ثم تحسين سرعته ثم مقارنة هذه النتائج التي تم الحصول عليها مع النتائج التي تم جمعها من خلال تطبيق العديد من الأساليب الأخرى. تثبت هذه المقارنات فعالية طرق البحث المقترحة في تقليل عدد التكرارات المطلوبة للحصول على مسار الروبوت. كما أنها قللت بشكل كبير من الوقت المطلوب لتشغيل البرنامج. من ناحية أخرى ، تم تحسين معدل سرعة الروبوت بنسبة تصل إلى 300٪ ، وهذا يشير إلى فعالية النهج الثاني المقترح..