2019

Performance evaluation of Max-Min Ant System Algorithm for Robot Path Planning in Grid Environment

Satya Shree Sankini
University of Windsor

Performance Evaluation of Max-Min Ant System Algorithm for Robot Path Planning in Grid Environment

by

Satya Shree Sankini

A Thesis
Submitted to the Faculty of Graduate Studies
through the School of Computer Science
in Partial Fulfillment of the Requirements for
the Degree of Master of Science at the
University of Windsor

Windsor, Ontario, Canada
2019

Performance Evaluation of Max-Min Ant System Algorithm for Robot Path Planning in Grid Environment

by

Satya Shree Sankini

APPROVED BY:

C. Chen,

Department of Electrical and Computer Engineering

I. Ahmad,

School of Computer Science

D. Wu, Advisor

School of Computer Science

May 15th, 2019.

Declaration of Originality

I hereby certify that I am the sole author of this thesis and that no part of this thesis has been published or submitted for publication.

I certify that, to the best of my knowledge, my thesis does not infringe upon anyones copyright nor violate any proprietary rights and that any ideas, techniques, quotations, or any other material from the work of other people included in my thesis, published or otherwise, are fully acknowledged in accordance with the standard referencing practices. Furthermore, to the extent that I have included copyrighted material that surpasses the bounds of fair dealing within the meaning of the Canada Copyright Act, I certify that I have obtained a written permission from the copyright owner(s) to include such material(s) in my thesis and have included copies of such copyright clearances to my appendix.

I declare that this is a true copy of my thesis, including any final revisions, as approved by my thesis committee and the Graduate Studies office, and that this thesis has not been submitted for a higher degree to any other University or Institution.

Abstract

Path planning is an essential task for the robot to navigate and control its motion in any environment. The optimal path needs to be rerouted each time a new obstacle appears in front of the robot in the dynamic environment. This research focuses on the MAX-MIN Ant System Algorithm(MMAS) which is an Ant Colony Algorithm derived from Ant System(AS) and is different from it in terms of the pheromone deposition. The effectiveness of this algorithm to obtain a near optimal solution is illustrated by the means of experimental study. Using a greedier search than the Ant System algorithm is one of the specific characteristics of the MMAS, which will be studied in the research. The robot environment model is represented by a grid which has obstacles whose positions change in each map that is used. Local search routines and diversification mechanisms introduced by the previous researchers are used to enhance the performance of the MMAS algorithm. To implement the MMAS algorithm used in our research, the experiments are performed in MATLAB development environment where a simulation program is designed, and the algorithm is implemented in grid maps of sizes starting from the smallest grid 10x10 to the grid of size 400x400. We implemented and analyzed the performance of the algorithm in larger grid environments to understand how it would perform when the search space is too huge; which would enable researchers to use the MMAS algorithm in experiments involving real life environments. In our experiments a new obstacle is added after every iteration of the algorithm which makes it challenging for the robots to find the near optimal path. The performance evaluation of the MMAS algorithm is studied and is also compared to that of the ACO algorithm when implemented in differently sized grid maps.

Dedication

I would like to dedicate this thesis to my family.

Father: Venugopal Sankini

Mother: Swarooparani Ennelli Sankini

Sister: Dhana Shree Sankini

Acknowledgements

I would like to express my sincere appreciation to my supervisor Dr. Dan Wu for his constant guidance and encouragement during my whole Master's period in the University of Windsor. Without his valuable help, this thesis would not have been possible. I would also like to express my appreciation to my thesis committee members Dr.Chunhong Chen, and Dr.Imran Ahmad. Thank you all for your valuable guidance and suggestions to this thesis. Last but not the least, I want to express my gratitude to my parents, my sister and my friends who gave me consistent help over the past two years.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Overview

Artificial intelligence (AI) is a branch of computer science that focuses on the creation of intelligent machines whose working, and the reaction is similar to that of humans [20]. With the increase in the development of computer technology, control theory, artificial intelligence theory, and sensor technology, robotic research has entered an entirely new phase [21]. One of the essential branches of robotics is mobile robots; it has received full recognition among the academicians around the world. Navigation in Encyclopedia Britannica is defined as the science of providing directions to a craft by determining its position, path, and the distance traveled [26]. For example, a craft finding its path to the desired destination while avoiding collisions is the technique of navigation [1]. Mobile robot navigation is thus the ability of a mobile robot to get from one place to another in an orderly manner without any human intervention to reach the targeted destination provided it has a perfect path planning system [1]. Path planning is defined as the determination of a path that a robot must take to navigate over each obstacle from the start to the goal position in an environment [25].

In all applications of mobile robots, they perform the navigation tasks using the following building blocks [20] in the Figure 1.1.

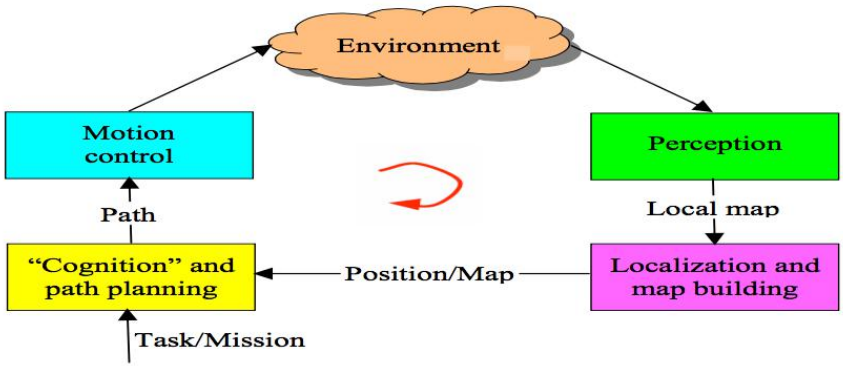

Figure 1.1: Building blocks of navigation [20].

It is evident from the Figure 1.1 that navigation of a mobile robot involves the perception of the environment, localization and map building, cognition and path planning and motion control. A lot of research has been done on path planning that is said to be one of the main components of the robotic navigation systems in distinct environments [5], [20], [26]. For a robot to successfully navigate, the environment has to be sensed by it to avoid dangerous circumstances and have path planning modules to steer towards a goal by itself.

1.2 Thesis Motivation

In the recent years, finding an optimal or a near optimal path(the final outcome) in robot navigation problem has been an interesting challenge as it involves considering factors such as the complexity of the environment in which the robot is placed, the parameters used, the efficiency of the algorithm applied to the problem. The factors mentioned previously can affect the final outcome. A lot of researchers have worked and are still working in this area to propose different methods or techniques to address the problem. Ant colony optimization algorithm(ACO) and its variants such as

the Max-Min Ant System(MMAS), Ant System Algorithm(AS); were introduced to solve the problem of path finding in robotics. The algorithms are usually applied to many combinatorial optimization problems. The main purpose of the combinatorial optimization (CO) problems is to find an optimal object from a finite set of objects [22]. Travelling Salesman Problem(TSP) is a combinatorial problem that has been extensively used by researchers in [10] [20] [14] [24] [16] to study the performance of ACO and its related algorithms.

MMAS was first introduced by Thomas Stutzle in the year 2000; it uses a greedier search compared to the traditional ACO algorithm [24]. In [24] the performance of the algorithm has been studied on the TSP problem. Later, the research on MMAS has progressed to studying the performance when implemented on a grid map in a static environment. On the contrary, in [21] MMAS has been implemented on a topological map and the paths that need to be traveled by the ants are represented by a sequence of actions that the ants should execute to reach the goal and the traveled distance spent by ants was analyzed. However, the performance of the MMAS algorithm was not evaluated when implemented on a large scale environment in both grid and topological maps. The grid map represents the robot environment(search space) even with minor details. Therefore, I found that as a motivation to further study the performance of the MMAS algorithm on a grid map(of sizes starting from 10x10 until 400x400) in a dynamic environment and contribute to the field of research.

1.3 Problem Statement

This thesis focuses on the performance evaluation of the Max-Min Ant System Algorithm(MMAS); a variant of the ACO algorithm in grid maps of smaller sizes such as 10x10, 20x20 and 40x40; and also on larger sizes of maps such as 100x100, 200x200 and 400x400 in a dynamic environment. The ACO algorithm is also implemented on

the same environments and the performance of the two algorithms is compared. We have designed the maps and implemented the algorithm in MATLAB environment. The changes such as adding diversification mechanisms and local search routines were made to the MMAS algorithm by the previous researchers; were also used in our experiments. The path length, the time taken by the algorithm to find a near-optimal path and the iteration at the which the path is first found and also the iteration at which the algorithm converges to a solution were recorded which helped us in analyzing the performance of the algorithm. The performance of the algorithm was analyzed using the best performance index, time performance index and the robustness performance index.

1.4 Thesis Organization

Chapter 2 introduces the key concepts of our research such as the mobile robot path planning, static vs dynamic environment, occupancy grid map. Later, the ACO algorithm is introduced and is discussed in detail. A description of how the behavior of the real ants inspires the ACO algorithms is given. Also, the three main types of ACO algorithms: Ant System, Ant Colony System, MAX-MIN Ant System are reviewed with more focus on the Max-Min Ant System Algorithm.

Chapter 3 explains the modeling of the robot motion environment using the grid method and introduces two of the main techniques that can be used for marking the grid. A path planning technique based on MMAS is explained and the technique of path re-routing is also discussed.

In chapter 4, the performance evaluation of the ant colony algorithms performed by other researchers are discussed. A brief description about the simulation environment and the necessary parameters is given. Based on the MATLAB, 6 comparative experiments are designed in dynamic grid environemnts to study the performance

of the improved MMAS algorithm compared to the ACO algorithm in different grid environments. The results obtained are given and an analysis of the obtained results is mentioned and discussed clearly.

Chapter 5 summarizes the work of this thesis. The conclusions obtained through our experiments are discussed and also the possible future work of our thesis is mentioned.

Chapter 2

Background Study

This chapter consists of all the parallel work used for the building of the key concepts and the methods used in our research study. In this chapter, we explain the key concepts of our thesis, we also discuss the Ant Colony Optimization Algorithm(ACO) and Max-Min Ant System Algorithm(MMAS) in detail.

2.1 Key Concepts

In this section, we will discuss the key concepts to understand the research work done in this thesis. Firstly, we will discuss the research area of the thesis and then we will describe what is Static and Dynamic environment, Introduction to Ant colony optimization and the Max-Min Ant System Algorithm.

2.1.1 Mobile Robot Path Planning

Mobile robots are commonly used in many industrial fields. Thus, the research on path planning for a mobile robot to avoid the obstacles in its path is very important. As discussed in Section 1.1, path planning enables the selection and identification of a suitable path for the robot to traverse in the workspace area [5]. There are two

kinds of path planning; namely the global path planning and the local path planning. The two main elements for global or deliberative path planning are [5]:

- Robot representation of the world in configuration space (C-space)
- Implementation of the algorithm

These two components are interrelated and significantly influence one another in the process to determine a near optimal route for the robot to traverse in the workspace within a reasonable amount of time [5]. If the information about the environment is known, the global path can be planned offline before the robot starts to move. This global path can assist the robot to traverse within the real environment because the attainable near optimal path has been formed within the environment. However, another category of path planning system known as local path planning was introduced; where the robot finds the near-optimal path while dealing with obstacles. The local path is usually constructed online when the robot avoids the obstructions in an environment [17].

Based on the environment and their functioning to find the near optimal path, the path planning algorithms can be classified into two categories , namely, local path planning algorithms and global path planning algorithms. The local path planning algorithms generate a path while the robot moves through the environment and can produce a new path based on the environmental changes. While global path planning algorithms need to have previous knowledge about the environment; therefore, the environment should be static.

While Path Planning is an important instance in the process of finding the near optimal path in an environment, localization holds an equally important role. Localization is the problem for determining a robot's position and orientation within the environment(that is deduced to a map). For example, you're in a room and browsing the internet at night. Suddenly the lights go off. You want to turn the torch on, but

your torch is in the bedroom closet, and its far away from the room where you are sitting. How will you reach there? The answer to the question is that; firstly, you need to stand up from your chair move towards the room while walking you use your hands as a sensor to sense where you are going. You touch the objects (landmarks) that you're already familiar with; like a door, wall, TV, etc., with your hands and find out your position(localize) and move towards the closet(destination) in the bed room. Therefore, we can say that by sensing the robot movements and the perceptions of the environment the localization problem can be solved.

2.1.2 Static Vs Dynamic Environment

The environment has a substantial impact on the difficulty of robot path planning. The environment used could either be static or dynamic. The environment in which only the position of the robot changes is known as the static environment. Comparatively, in dynamic environments, the objects other than the robot are present whose location or configuration changes over time. Most real environments are dynamic, where the position changes of the objects present occur at a different range of speeds.

Our idea of the dynamic grid map which is an occupancy grid map is that a new obstacle is introduced at every iteration of the algorithm; it means that a new obstacle is added or the position of the existing obstacles is altered after every cycle of the implemented algorithm within the given value for the number of iterations. This would make the environment completely new for the robot to traverse through every time the algorithm is initiated because there is a new obstacle added.

We discuss the occupancy grid map in the Section 2.1.3

2.1.3 Occupancy Grid Map

The occupancy grid map is represented as a field of random variables in an evenly spaced grid [27]. A value is assigned to each random variable that represents its

occupancy. The value of the variable can be one of the three:

- **Free:** Space has been explored, and the robot knows it is free of obstacles.
- **Occupied:** There are obstacles present in the grid and have been sensed by the robot.
- **Unknown:** The grid has not been explored and no idea about its occupancy.

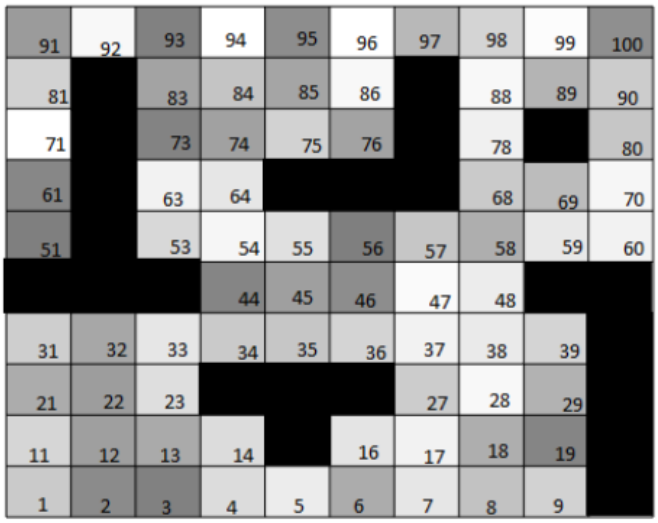

Figure 2.1: Occupancy numbered grid map.

The Figure 2.1 is the representation of the occupancy grid map. In the figure it is clear that each grid is numbered starting from 1 to 100, the black objects represent the obstacles that have occupied the grids.

2.2 Introduction to Ant Colony Optimization

Ant Colony Optimization (ACO) was first introduced and proposed by Marco Dorigo in his Ph.D. thesis in 1992 [6]. It was inspired by the food-seeking behavior of the ant colony, and it is said to be a metaheuristic. According to Dorigo, the behavior of a single ant seems quite simple. However, he also mentioned in his research that

multiple ants can cooperate to form an enormous social group to accomplish many other complicated tasks. The ants initially wander randomly to explore the environment near their nest, while searching for food. Different paths can be chosen by different ants to explore in compliance with their random behavior [6]. As soon as a food source is determined by an ant; it carries a bearable amount of food on its way back to its nest.

By a lot of research and study; biologists have found that ants leave some chemical substance on the paths that were traversed by them, which is called pheromone, and the quantity of pheromone is inversely proportional to the length of the route. It was also mentioned in Dorigo's research in 1992 that the ants can also perceive the pheromone when they pass the path, and their actions could be influenced by the concentration of pheromone. However, the amount of the pheromone left may vary depending on the quality and quantity of the food. Then, other ants now can choose the route that has denser pheromone (which can be treated as a better route) and guides them to the food source. This behavior will also cause the formerly better path to becoming even much better by aggregating more pheromones. This behavior of the ants is depicted in Figure 2.2.

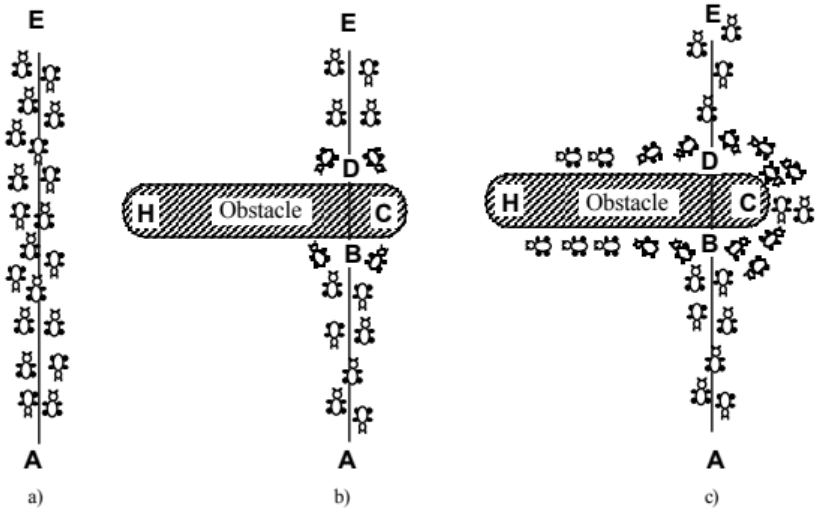

Figure 2.2: Illustration of the behavior of the ants [10].

We can see that the nest of the ants in Figure 2.2 is denoted by **A** and the food source is denoted by **E**. In Figure 2.2 (a), (b) and (c), the space between the starting (**A**) and the goal positions(**E**) is the environment that is to be explored by the ants. In Figure 2.2(a) the ants move freely between the points **A** and **E**.

However, in Figure 2.2 (b) an obstacle is placed unevenly to cut off the path between the points **A** and **E**. The longer part of the obstacle is denoted by **H**, and the shorter part is denoted by **C**. Once the obstacle is added the ants have to now choose between the paths **AC** or **AH** to reach the food source **E**. At first, the ants randomly choose either of the paths **AH** or **AC**. After a considerable amount of time, the ants realize that taking the path **AC** makes them reach the point **E** faster because that is the shortest path. Therefore, more number of ants follow this path and as they leave some amount of pheromone while traversing, the concentration of the pheromone on the path **AC** is higher compared to that on the path **AH**. This eventually leads to all the ants taking the path **ACE** than the path **AHE** making the concentration of pheromone on the former path denser.

This behavior of the ant colony is termed as metaheuristic, which is said to be the main concept of the ant colony optimization which falls under the category of the approximate algorithms. The approximate algorithms are used to obtain the near-optimal solutions to solve hard combinatorial optimization problems (CO).

2.2.1 The ACO Metaheuristic

ACO has been formalized into a metaheuristic for combinatorial optimization problems by Dorigo et.al in the year 1992 and 1996. A metaheuristic is a general-purpose algorithmic framework that can be applied to different optimization problems with a few modifications. For example, In order to apply ACO to a given a combinatorial optimization problem we would need an adequate model [27],

Here we consider that a model $\boldsymbol{P}$ consists of:

- A search space $\boldsymbol{S}$ defined over a finite set of discrete decision variables X_i, where $i = 1, 2, ...n$.
- A set of Ω constraints among the variables.
- An objective function $f : S \rightarrow R_0^+$ to be minimized.

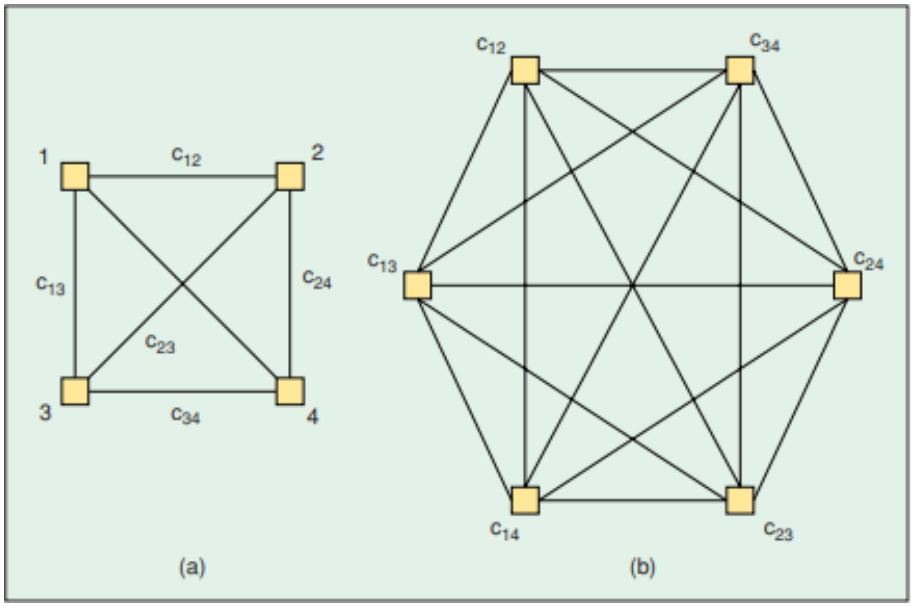

Figure 2.3: Example of possible construction graphs for a four-city TSP where components are associated with (a) the edges or with (b) the vertices of the graph [27].

Figure 2.3 shows the example of possible construction graphs for a four city TSP problem, as depicted by the researchers in [27]. In the Figure 2.3 (a), the components of the TSP are associated with edges that means $1, 2, 3, 4$ are the vertices(i.e, cities) and $c_{12}, c_{24}, c_{34}, c_{13}, c_{23}, c_{14}$ are the edges that show the relationship between the vertices. In the Figure 2.3 (b), the components are associated with the vertices of the graph; which means that the edges are considered as nodes represented by $c_{12}, ..., c_{14}$ and the lines connecting the nodes show the relation between the vertices. When a CO problem has to be solved, a finite set of solution components C are to be derived. Each possible solution components are associated to a pheromone value τ. Initially,

pheromone value τ_{ij} is associated with the solution component c_{ij}; where i and j are the cities.

One of the essential components of the ACO metaheuristic is the pheromone model. In ACO, by traversing the fully connected graph $G_C(V, E)$; where V is the set of vertices and E is a set of edges. The mentioned connected graph G_C can be obtained by representing the solution components C either by vertices or by edges. A partial solution is built when the ants move from one vertex to vertex along the edges of the graph. A certain amount of pheromone is deposited by the ants on the components traversed(either on the vertices or the edges). However, the amount of $\triangle\tau$ is dependent of the quality of the solution found. Therefore, the successive ants use the pheromone information to plan their path on the most favourable regions of search space i.e the graph. Usually, a set of n variables represent a solution; where a city is associated with each variable. For example, X_i is a variable which indicates the city to be visited after i. The pairs of cities to be visited one after the another are represented by the solution components $c_{ij} = (i, j)$ where, i and j are the cities; which means that the city j should be immediately visited after city i . Therefore, in this scenario of the construction graph the vertices are the cities to be traversed and the edges are the solution components. Therefore, the ants would deposit the pheromone on the edges. However, a construction graph could also be obtained by representing solution components as vertices on which the pheromone is deposited. This method of obtaining a construction graph is not very popular but is nonetheless correct [16].

The Figure 2.4 is an ACO metaheuristic. Once initialized, the algorithm iterates over three phases: solutions are constructed at each phase by the ants, a local search is applied to improve the solutions after which the pheromone is updated[27].

Algorithm 1 Ant colony optimization metaheuristic

```
Set parameters, initialize pheromone trails
while termination conditions not met do
    ConstructAntSolutions
    ApplyLocalSearch        {optional}
    UpdatePheromones
end while
```

Figure 2.4: ACO Algorithm [2].

The three main algorithmic components: ConstructAntSolutions, ApplyLocalSearch, UpdatePheromones are explained further in detail.

- **ConstructAntSolutions**

  Artificial ants can be viewed as probabilistic useful methods that collect arrangements as progressions of parts of the arrangement. The limited arrangement of finite set of the solution components $C = c_{ij}$ where $i = 1, ..., n$ and $j = 1, ... \mid D_i \mid$ [2]. A solution construction commences from an empty partial solution $S_p = \emptyset$. At every construction step, the partial solution S_p is elongated by adding a feasible solution component from the set $N(S_p) \subseteq C$, which is defined as the set of components that can be appended to the current partial solution S_p without violating any of the constraint values. The determination of a solution component from $N(S_p)$ is supervised by a stochastic mechanism, which is biased by the pheromone correlated with each of the elements of $N(S_p)$.

- **ApplyLocalSearch**

  LocalSearch is usually included in state-of-the-art ACO algorithms [6]. After all the ants finished the partial solution construction in one iteration, the pheromone should be updated to increase its value to keep a strong association with the good or promising solutions. There are two main steps:

  – Decrease all the pheromone values through pheromone evaporation.

  - Increase the pheromone levels associated with a chosen set of good solutions.

  Once solutions have been built, and before refreshing the pheromone, it is common to augment the solutions obtained by the ants by a local search. This phase, which is exceptionally problem-specific, is optional although it is usually included in state-of-the-art ACO algorithms.

- **UpdatePheromones**

  The pheromone update aims to raise the pheromone values correlated with beneficial or promising solutions and to reduce those that are associated with poor ones. Usually, this is obtained (i) by lowering all the pheromone values through pheromone evaporation, and (ii) by raising the pheromone levels associated with a chosen set of good solutions.

2.2.2 Main ACO Algorithms

A lot of research on the ACO algorithms and their implementation has been done previously; such as in [23], [6], [11], [21] and [24] among the others. The main ACO algorithms are:

- Ant System Algorithm (AS)
- Ant Colony System Algorithm (ACS)
- Max-Min Ant System Algorithm (MMAS)

Further in this section, we will discuss the implementation of three main algorithms of the Ant Colony Optimization on the Travelling Salesman Problem; with more emphasis on the MMAS algorithm as it is the basis of our thesis.

Ant System Algorithm

Ant System is the first ACO algorithm proposed in the literature [7] and [9]. The main characteristic of this algorithm is that the pheromone values are updated by all the m ants that have built a solution, at each iteration.

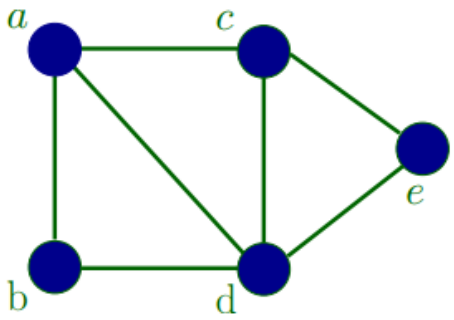

Figure 2.5: Construction Graph.

For example, assume the Ant System as a connected construction graph consisting of vertices and edges as shown by Figure 2.5; represents a 5 city TSP problem. In the figure 2.5 the nodes a,b,c,d,e are the cities and the lines connecting them are the edges. Here, for a vertex pair (ab) and edge (a,b), the pheromone would be denoted by τ_{ab}, associated with the edge joining cities a and b, is updated as follows:

$$\tau_{ab} \leftarrow (1-\rho).\tau_{ab} + \Sigma_{k=1}^{m} \triangle \tau_{ab}^{k} \tag{2.1}$$

where ρ is the evaporation rate, m is the number of ants, and the quantity of pheromone deposited on edge (a, b) by ant k which is denoted by $\triangle\tau_{ab}^{k}$. It can be represented by the equation (2.2) where Q is a constant, and L_k is the length of the tour constructed by ant k in the present iteration.

$$\tau_{ab}^{k} = \begin{cases} \dfrac{Q}{L_k}, & \text{if ant k used edge (a,b) in its tour} \\ 0, & \text{otherwise} \end{cases} \tag{2.2}$$

By having the τ_{ab} value, the probability for ant k to go to vertex b from a is calculated as shown in the equation (2.3).

$$P_{ab}^{k} = \begin{cases} \frac{\tau_{ab}^{\alpha}.\eta_{ab}^{\beta}}{\Sigma_{C_{al} \in N(s^p)}.\tau_{al}^{\alpha}.\eta_{al}^{\beta}} & if C_{ab} \in N(s^P), \\ 0 & \text{otherwise,} \end{cases} \tag{2.3}$$

In equation (2.3) the first condition is satisfied only if C_{ab} belongs to the set of possible components which is here denoted by $N(S_p)$. It means that vertices (a,l) where l is a city not yet visited by the ant k. The relative effect of the pheromone versus the heuristic information η_{ab} is controlled by the parameters α and β, which is given by the equation (2.4), where d_{ab} is the distance between cities a and b,

$$\eta_{ab} = \frac{1}{d_{ab}} \tag{2.4}$$

Similar to which the process was described in theoretical explanation in Section 2.2.1, in each iteration, based on the paths traveled by each ant, the pheromone τ_{ab} on each edge of the connected construction graph is updated using equation (2.2). Ants will traverse through the graph probabilistically choosing next vertex by using equation (2.3) when the next round of iteration begins, based on the updated pheromone.

Ant Colony System

The Ant Colony System (ACS) was first introduced in 1996 [8]. The same author has proposed the concept of ant colony optimization in his Ph.D. thesis in 1992 [6]. A local pheromone update was added to the pheromone update performed at the end of the process of construction of a solution.

According to [13], the three main aspects in which the ACS differs from the Ant System are:

i) A direct way to balance between the exploration of new edges and exploitation of a priori and registered knowledge on the problem is provided by the state transition rule.

ii) The global updating rule is applied only to edges which relate to the best ant tour.

iii) A local pheromone updating rule (local updating rule, for short) is applied while ants are constructing a solution. The quantity of pheromone on the traversed edges is updated by the ants while constructing their tour using the local updating rule. The pheromone updating rules are outlined so that they tend to give more pheromone to edges which should be visited by ants [13].

2.3 MAX-MIN Ant System

The Max-Min algorithm was first proposed in the year 2000 by T. Stutzle and H.H. Hoos. Many studies were conducted on ACO [4], [23], [26] which have shown that by stronger exploitation of the best solutions (obtained during the search and the exploration of the environment) the performance of the algorithm could be improved. However, in [24] the author has introduced the concept of using a greedier search in the MMAS algorithm; the introduced concept potentially aggravates the problem of premature stagnation of the search. MAX-MIN Ant System differs in three key aspects from the Ant System, namely:

- Only one single ant adds pheromone after each iteration. That is done to exploit the best solutions found during an iteration or the run of the algorithm [24]. This ant could be the one who found the best solution in the current iteration (iteration-best ant) or the one who found the best solution from the beginning

of the trial (global-best ant).

- The range of possible pheromone trails on each solution component is limited to an interval $[\tau_{min}, \tau_{max}]$, to avoid stagnation of the search [24].
- The pheromone trails are initialized to τ_{max}, to achieve a higher exploration of solutions at the beginning of the algorithm [24].

Pheromone Trial Updating

The modified pheromone trail update rule is given by

$$\tau_{ij}(t+1) = \rho\tau_{ij}(t) + \triangle\tau_{ij}^{best} \tag{2.5}$$

where $[\Delta\tau]_{ij}^{best} = 1/(L_{best})$; here, L_{best} is the length of the tour of the best ant. It could either be the best tour found in the present iterationiteration-best, L_{ib} or the best solution found since the initialization of the algorithm L_{bs} (best so far) or a combination of both.

The notion of convergence for MAX-MIN Ant System which is needed in determining the values of pheromone trial limits was introduced in [24]. The concept of convergence of MMAS differs in one slight but important aspect from the concept of stagnation [10]. All ants follow the same path in stagnation. However, in situations of convergence, it is not the similar case due to the use of the pheromone trail limits.

Pheromone trail initialization

In MMAS the pheromone trails are initialized in such a way that after the first iteration all pheromone trails correspond to $\tau_{max}(1)$. The strategy of all the pheromone trials corresponding to $\tau_{max}(1)$ can easily be achieved by setting $\tau(0)$ to some arbitrarily high value. The trails will be forced to register values within the set limits, after the first iteration of MMAS in particular, they will be set to $\tau_{max}(1)$. During the

first iterations of the algorithm, this type of trail initialization is chosen to increase the exploration of solutions.

2.4 Changes made to the MMAS Algorithm

2.4.1 Addition of Diversification mechanisms[24]

The diversification mechanism is used and to check if that allows MMAS to converge to a very high-quality solution. Two variants which contrast in the degree of search diversification are examined. Firstly, the pheromone trials are reinitialized to τ_{max} as given below,

$$\tau^*_{ij}(t) = \tau_{ij}(t) + \delta(\tau_{max}(t) - \tau_{ij}(t)) \tag{2.6}$$

However, this corresponds to setting $\delta = 1$ in the equation above whenever the pheromone trail strengths on almost all paths are not contained in S_{gb}(global best solution) are very close to τ_{min} (minimun pheromone value).

Once the algorithm is reinitialized, the search f_{gb} (iteration best solution) is applied as done at the start of the algorithm. The best solution found since the reinitialization of the pheromone trials is used instead of S_{gb}, by doing this more search diversification is obtained. The Max-Min Ant System algorithm is then allowed to converge to another high-quality solution.

2.4.2 Addition of local search routines[24]

Local search algorithms commence from a complete initial solution and try to find a better solution in an agreeably defined neighborhood of the current solution. In its most basic version, known as the iterative improvement, the algorithm searches the

neighborhood for an correcting solution. If such a solution is found, it displaces the current solution, and the local search resumes. These steps recur until no improving neighbor solution can be attained in the neighborhood of the current solution and the algorithm ceases in a local optimum (the best solution in among the available solutions in a neighborhood in an environment).

2.5 Literature Review and Related Work

In [10] the authors proposed the ant system as a new approach to stochastic combinatorial optimization. They have studied the performance of the ACO algorithm on a TSP problem in a static envirnment. Their basic idea was that if at a given point an agent (ant) has to choose between different options for its next move and the one actually chosen results to be good, then in the future that choice will appear more desirable than it was before. Also, the authors in [10] have shown that the Ant System problem could be applied to different CO problems.

The authors in [10] introduced the ACO meta heuristic algorithm. The ACO meta heuristic was implemented on a TSP problem in a static environment. They have also discussed the importance of the trial visibility and Trial persistence.

Later, in [24] the authors have introduced the MMAS algorithm. The algorithm's performance was evaluated when implemented o a TSP problem. The results obtained indicated that the performance of the algorithm was better when compared to the ACO algorithm as the former one uses a greedier search.

In addition to the research that has been discussed in the above paragraphs, the authors in [21] have implemented the MMAS algorithm on an evolutionary computation problem in which the robots should explore the environment at the same time they plan the path. The authors have used a Topological Map to represent their environment. The proposed MMAS algorithm provided a very good performance in

relation to a genetic algorithm.

Chapter 3

Environment Modeling and the Working of MMAS

3.1 Modeling of Robot Motion Environment

Autonomous navigation describes a higher level of performance since it applies obstacle avoidance simultaneously with the robot steering toward a given target [4]. Therefore, it implies an environment with known and unknown obstacles, and it incorporates global path planning algorithms [3] to organize the robots path amidst the known obstacles, as well as local path planning for real-time obstacle escape.

One standard technique for map representation that does not bear from data associations is to utilize occupancy grid maps to approximate the environment. An occupancy grid map depicts the environment as a block of cells, each one either seized so that the robot cannot pass through it, or abandoned, so that the robot can traverse it. Grid method is used to establish the environmental model to simulate the actual working area of the robot which in turn helps to avoid the complex calculations that may arise otherwise when dealing with the boundaries of the obstacle. In the application of the grid method, the division of the grid size is critical. The grid maps

we used in our experiments are the occupancy grid maps.

An initial implementation of occupancy grid maps was used by Moravec in his research in 1988 to automatically prepare a map of the environment. Sensor readings were matched to the map, altering the possibility that observed cells are filled. For example, a sonar sensor returns the nearest object within a cone, so the cells in the extent of the cone closer than the reading are probably unoccupied. Moravec in 1988 represented each cell in his reasearch as a possibility of being traversable and initializes them to an obscure value. He illustrated a probabilistic technique to update cells for several types of sensors and supplied a procedure to permit the map to be renewed as the robot moves [15].

According to Milstein's research in 2008 in [18], to build an occupancy grid map, it is required to determine the occupancy probability of each cell. Although, assumed that it is not surely accurate, especially when acknowledging nearby cells representing the equivalent physical object. As a result, the probability of a distinct map m, can be factored into the product of the different probabilities of it's cells.

The probability of a precise cell is simple to ascertain, given the robots location and sensor readings since it is defined by whether the robot observes the cell as unoccupied or occupied [18]. The Occupancy grid mapping refreshes a map according to a sensor reading at a location so that, as evidence accrues, the map becomes accurate.

Milstein in his research in [18], mentioned an example where we consider each cell of the map to be an independent object, which can be either present or absent. Although independence is usually not entirely valid, it was assumed to be so.

The assumptions can therefore be summarized as follows:

(1) 2D definite space is the environment in which the mobile robot moves.

(2) The movable trajectories of the dynamic objects can be estimated for the future, keeping the speed of the robot constant.

(3) Every time the robot makes a move it is directed to the grid center. The

environment information can be centered in the current grid center [18].

3.1.1 Robot movement in grid environment

In a grid environment, the number of obstacles are lesser in number in some instances compared to the others depending on the experiments being constructed (it is a decision to be made by the researcher.

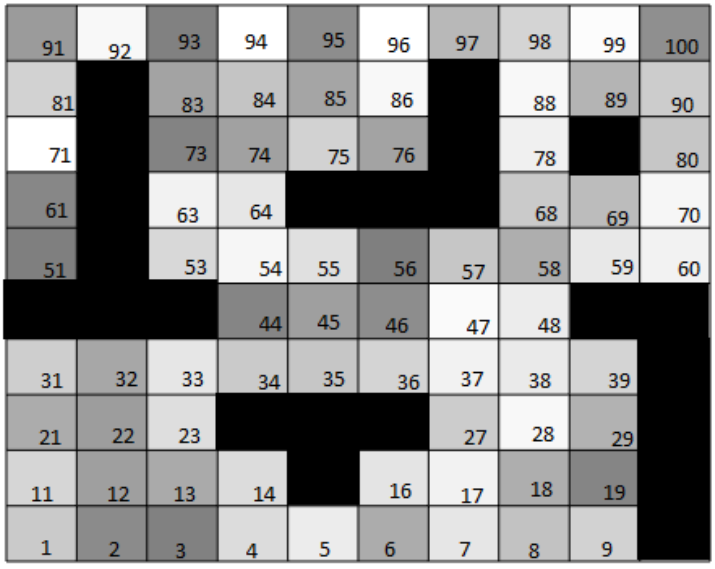

Figure 3.1: An example of the grid map.

For example, the Figure 3.1 is a two dimensional workspace with obstacles. The workspace would be divided into rows and columns composed of grids which are of equal size.The black grids denote the obstacles, and other white grids denote the free grids. Therefore, this method is based on the concept of occupancy grids. However, the size and the location of the obstacles present are unknown. The starting point of the algorithm implemented is set to the bottom left grid of the workspace. The grid cells are numbered from left to right starting from the bottom left grid which is numbered as *1* and the destination grid that is the top right grid is numbered as *100*.

The robot aims to search an optimal or near optimal path from the start to the destination grid (goal). We adopt two main methods for grid marking, namely the Rectangular coordinate method and the Serial number method.

- **Serial number method:** From the bottom left of the grid map, coding the grid from bottom to top, from left to right as shown in Figure 3.1. The serial number is from 1 to 100. The grid using the serial number method to mark is g_n, e.g., the grid of serial number 1 is marked as g_1.

- **Rectangular coordinate method:** The method in which every grid coordinate is indicated with central point (x, y). The grid using Rectangular coordinate method is $g(x, y)$, e.g., the grid of serial number 1 is marked as $g(0.5, 0.5)$. The starting point of robot path planning is assigned as g_1 in the bottom left of the map, likewise, the target point of robot path planning as g_{100}.

To simulate the real ant colony seeking food behavior in the given example in the Figure 3.1, we assume that the starting point of robot path planning g_1, and the target point g_n as a food source.

3.2 Theoretical Explanation of Path Planning

Once the modeling of the environment and the process of marking the grid is done, we would initialize the robot at grid g_1 and wait for it to navigate itself through the environment and reach the grid g_{100}.As discussed in Section 1 Navigation is a methodology that allows guiding a mobile robot to achieve a mission through an environment with obstacles healthily and safely. The two basic tasks included in navigation are the localization, and path planning.

According to the survey by Nirmala et.al in 2016, it can be said that the strategy used to find the solution consists of the two operations such as the recognition of a set of navigation and operation goals [19].

In the experiments implemented in this thesis, we have used grid maps with obstacles. The ants have a planned path from the starting to the goal position. However, once the algorithm is implemented, and a new obstacle is added everytime, in a while the agents would have to re-route their path from their current position which is assumed to be the start position after a new obstacle is added, and then a path is planned from that point to the goal position.

This process is repeated every time a new obstacle is added. The Figures 3.2, 3.3, 3.4 depict the way the robots plan their path. In these figures the black and the grey objects represent the obstacles, the red line represents the path travelled by the ants , the dotted red line represents the previous path travelled by the ants; the grid on the bottom most left is the starting position of the ants and the grid on the top most right is the goal position. The process of path planning and path rerouting is further explained in detail.

In the Figure 3.2 the ants is placed at the starting position. A few obstacles are already present in the environment. Once the algorithm is run, the robot starts following the path towards it's destination which here is represented by the red line.

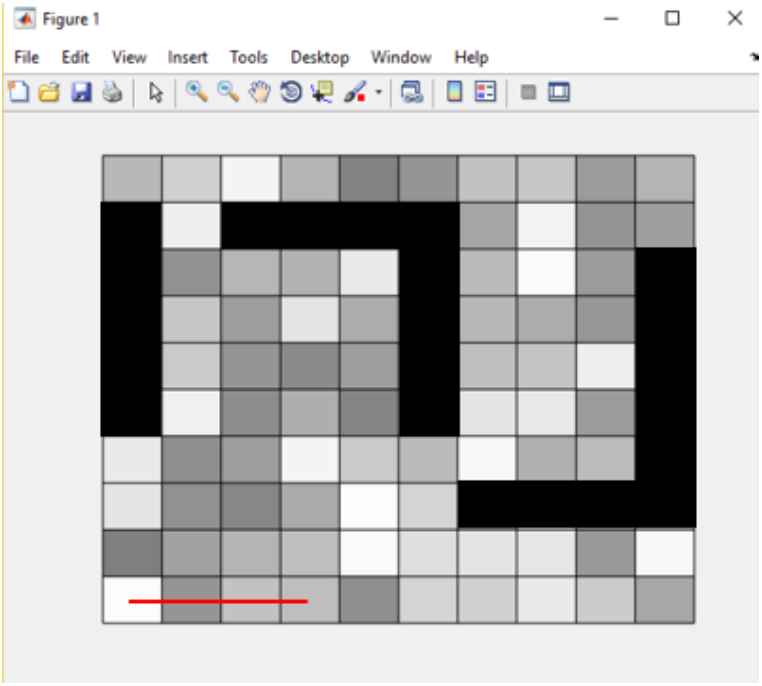

Figure 3.2: The example of path re-routing(a).

In the Figure 3.3 the black coloured blocks represent the newly added obstacles, the grey coloured blocks represent the obstacles that were previously added. The robot has to analyze and plan it's path from the grid that it was present in which is considered as the starting point after the new obstacles are added. The robot moves from that point towards its goal position following a totally different path that it had planned earlier before the obstacles were added.

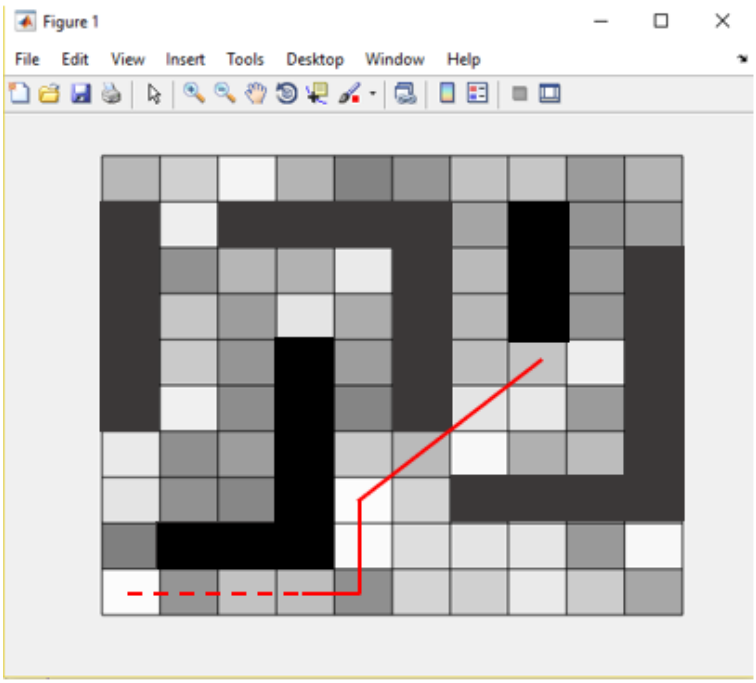

Figure 3.3: The example of path re-routing(b).

The ants finally reach the destination grid after rerouting their path every time they encounter an obstacle. This is shown in the Figure 3.4.

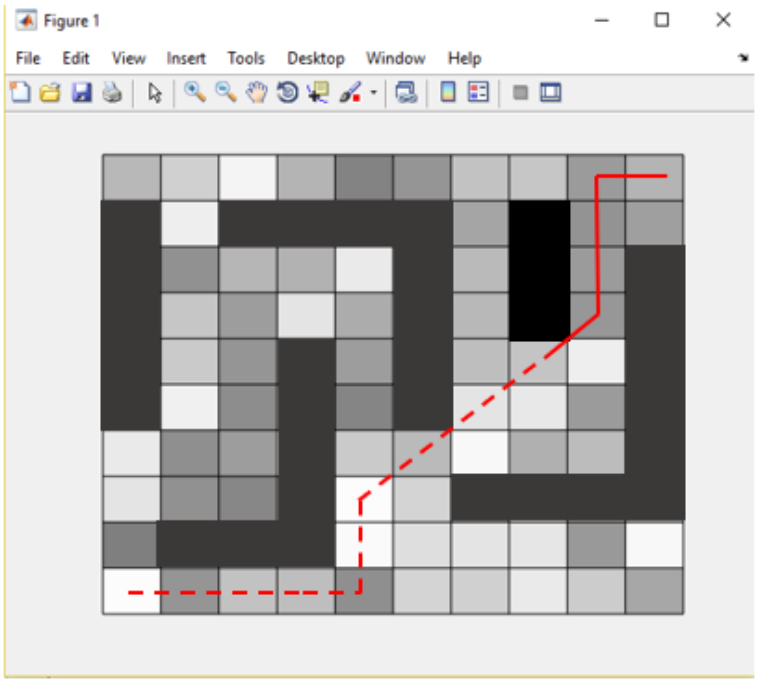

Figure 3.4: The example of path re-routing(c).

3.3 Robot Path Planning Based on Max-Min Algorithm

In the research of the MMAS algorithm, it has been found that the algorithm solves the TSP problem successfully. But before applying MMAS on the field of robot path planning, many changes to the traditional algorithm have to be made based on the features of robot path planning. It introduces τ_{max} (upper) and τ_{min} (lower) bounds to the values of the pheromone trails, as well as a distinctive initialization of their values.

In practice, the permitted range of the pheromone trail strength is limited to the interval that is $[\tau_{min}, \tau_{max}]$, and the pheromone trails are initialized to the upper trail limit, which induces a higher exploration at the start of the algorithm [24]. The MMAS algorithm was modified by Stuzle et.al in the year 2000 after he introduced the algorithm in the year 1999, to achieve faster convergence speeds while still finding shorter paths.

The experimental results are given in chapter 4 and conclusion in chapter 5.

Chapter 4

Analysis, Evaluation of Results and Discussion

In this chapter, we shall discuss the experiments conducted in detail. Also we have summarized and analyzed all of the results obtained.

4.1 Simulation

All the experiments were conducted on a PC with device specifications a 3.40GHz Intel CORE i7-6700 processor, 8 GB of RAM and 64-bit Windows operating system.

All the simulation programs have been compiled using MATLAB.

4.2 Parameter Settings

- **The number of ants m:**

  The number of ants m has an important influence on the overall performance of the ant colony optimization algorithm. In general, the capability of the algorithm could be increased using a certain number of ants. However, if the number of ants m is oversize, the pheromone variation would be reduced to average and

the convergence speed is slowed down, on the contrary, if the number of ants m is too few, the stability of the algorithm is reduced, and the problem of early stagnation would occur [12]. In our thesis we have set the number of ants as 20 for the grids 10x10, 20x20 and 40x40. However, the number of ants required to implement the algorithms for the grids 100x100, 200x200 and 400x400 had to be more than 20 since the grid maps are of large sizes an it would help in better exploration of the environment and converge to a near optimal solution in feasible amount of time. However, when we initially used just 20 ants for the larger grid environments, we came across few problems which will discussed in the Section 4.5.

- **The combination of parameters α, β, ρ:**

  The relative importance of pheromone accumulated by ant colony is reflected by α which is the impact index of pheromones, and the relative importance of the heuristic information is reflected by β which is the impact index of a heuristic factor.

  The pheromone evaporation rate ρ reflects the intensity of the interaction between ants, which is directly related to the global search ability and convergence speed of the ACO and related algorithms. The global search ability of the MMAS algorithm can be improved by increasing the value of ρ. However, the convergence speed of the algorithm is reduced.

  As a matter of fact, the roles of α, β and ρ are closely related. While applying the Ant colony optimization algorithms on robot path planning, the wrong combination setting of α, β and ρ will eventually slowdown the solution speed and the quality of the results would be degraded. In our thesis we have used the following values for all the experiments: $\alpha = 5$, $\beta = 5$ and $\rho = 0.5$.

- **The pheromone values τ_{max} and τ_{min}:**

  In MMAS, we set the maximum pheromone trail τ_{max} to an estimate of the asymptotically maximum value.

  To determine reasonable values for τ_{min} , we use the following assumptions as mentioned in [24],

  - The best solutions are found shortly before search stagnation occurs. In such a situation the probability of re-constructing the global-best solution in one algorithm iteration is significantly higher than zero. Better solutions may be found close to the best solution found.
  - The main influence on the solution construction is determined by the relative difference between upper and lower pheromone trail limits, rather than by the relative differences of the heuristic information.

  Stutzle and Hoos have introduced a formula to calculate the maximum and the minimum trial limit values which are assumed to be approximate [24].

$$\tau_{min} = \tau max. \frac{1 - (P_{best})^{-(1-n)}}{avg - 1(P_{best})^{-(1-n)}} \tag{4.1}$$

  where avg is the average number of components that can be chosen in construction steps, P_{best} is the probability of constructing the best solutions, and n represents the number of components in the constructed solution.

- **Pheromone Intensity Q:**

  Pheromone intensity Q is the total amount of pheromone released by the ant colony left on the paths they traveled after a single iteration. The larger the Q, the faster the pheromone accumulation on the paths of the ant colony, the convergence speed of the algorithm is improved [28]. However, when Q is oversize,

the algorithm easily fall into a local optima.

- **The number of iteration N_c:**

  To ensure the algorithm can search the optimal path within the number of iterations, the value of N_c should be set larger. In this thesis, under the 10*10 grid environment map, N_c = 100, under the 20*20 grid environment map, N_c = 400, under the 40*40 grid environment map, N_c = 1600, under the 100*100 grid environment map, N_c = 10000, under the 200*200 grid environment map, N_c = 40000, under the 400*400 grid environment map, N_c = 160000.

4.3 The Implementation of Simulation Experiments

The aim of the thesis as mentioned in Section 1.3 is to implement the improved MMAS algorithm in larger dynamic grid maps. However, we have not only implemented the MMAS algorithm in a smaller grid environment but also on a larger environment to check for its scalability. Obstacles are set in every map whose position is unknown to the robot The experiments are later performed on 6-different sizes of the grid maps. The performance of the MMAS is also studied and compared with the ACO algorithm on different maps implemented by the previous researchers.

4.3.1 Collection of Maps Used in the Experiments

In this section we showcase the grid maps used in all of our experiments.

.

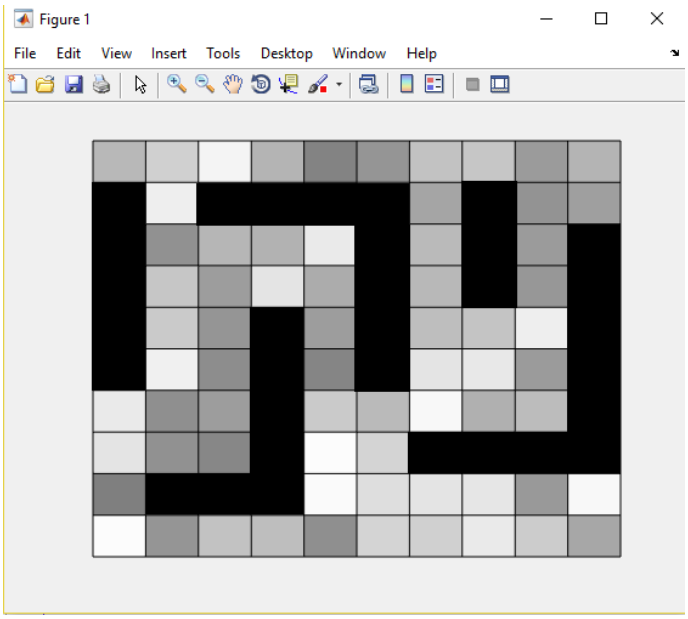

Figure 4.1: Map1: 10*10 Grid environment.

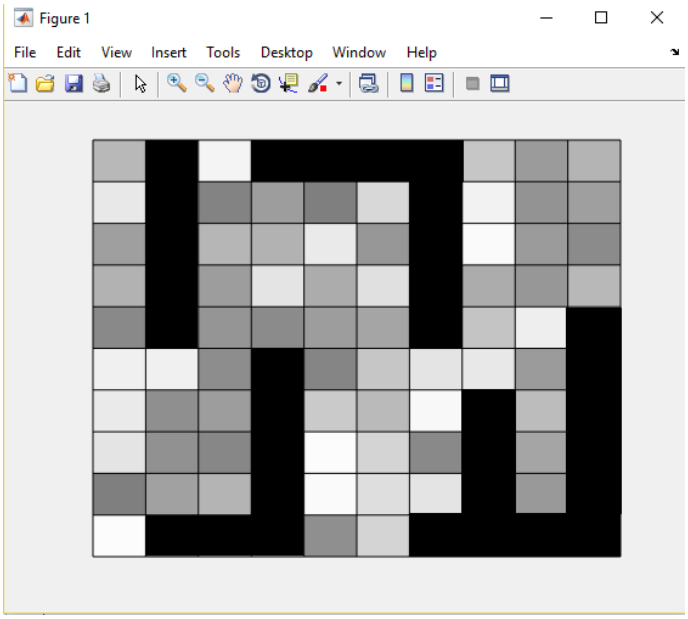

Figure 4.2: Map2: 10*10 Grid environment.

The Figures 4.1 and 4.2 represent the 10*10 grid maps

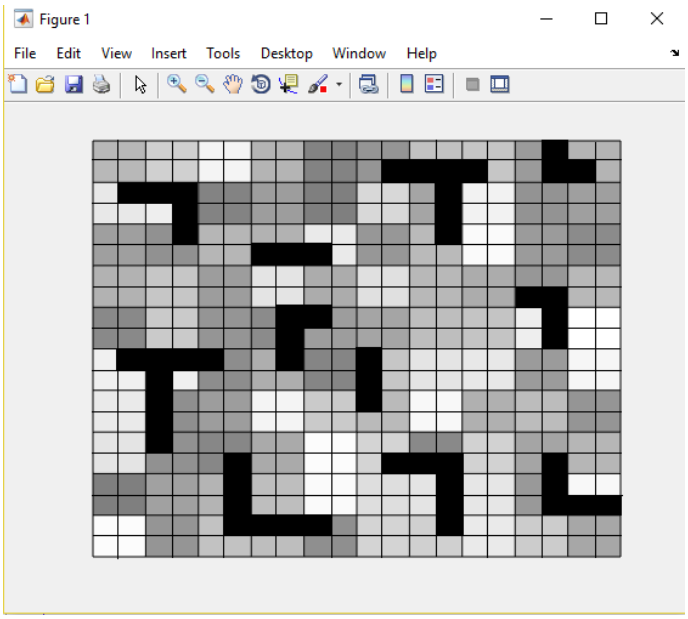

Figure 4.3: Map2: 20*20 Grid environment.

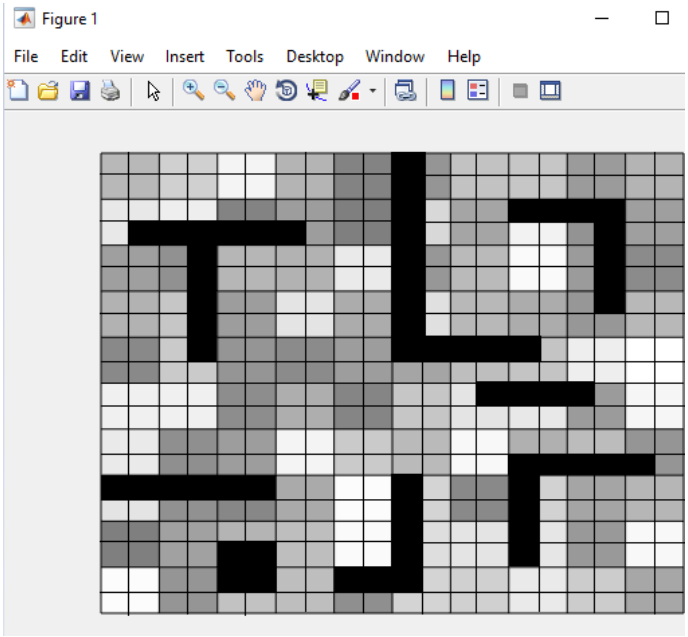

Figure 4.4: Map2: 20*20 Grid environment.

The Figures 4.3 and 4.4 represent the 20*20 grid maps.

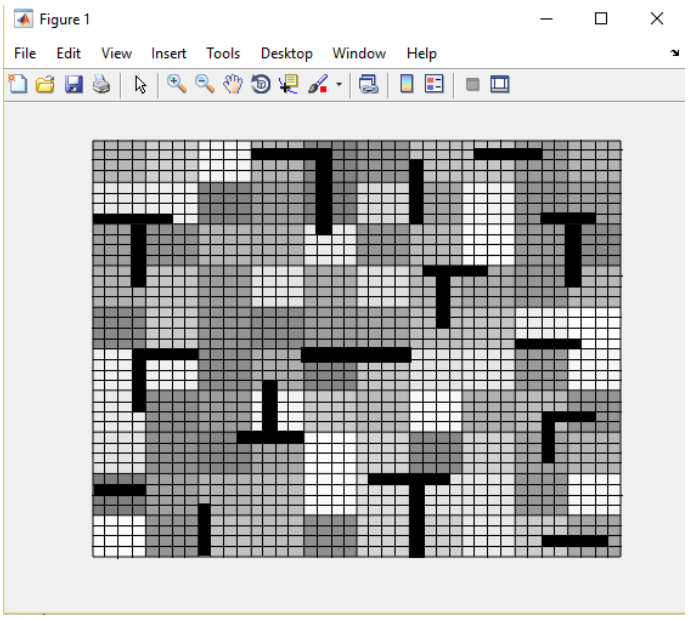

Figure 4.5: Map1: 40*40 Grid environment.

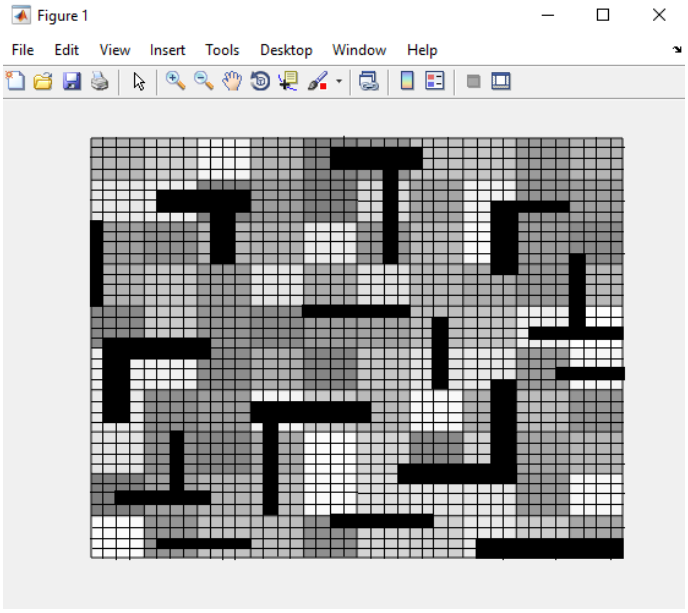

Figure 4.6: Map2: 40*40 Grid environment.

The Figures 4.5 and 4.6 represent the 40*40 grid maps.

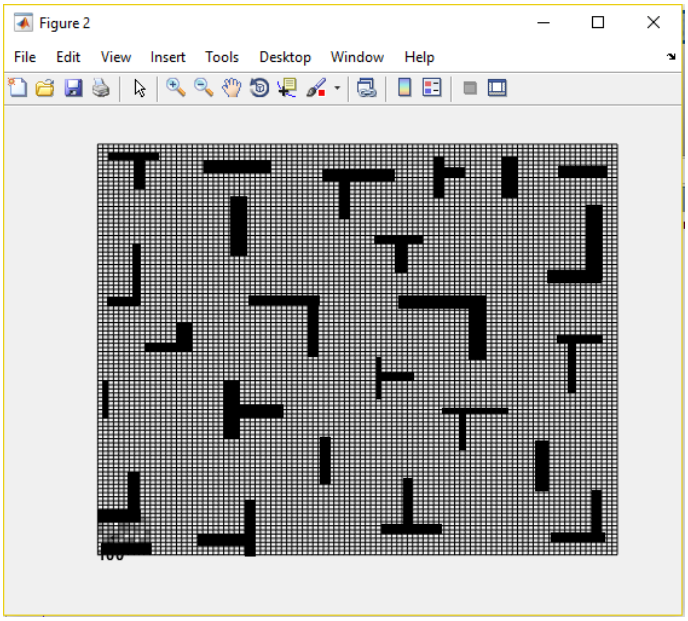

Figure 4.7: Map1: 100*100 Grid environment.

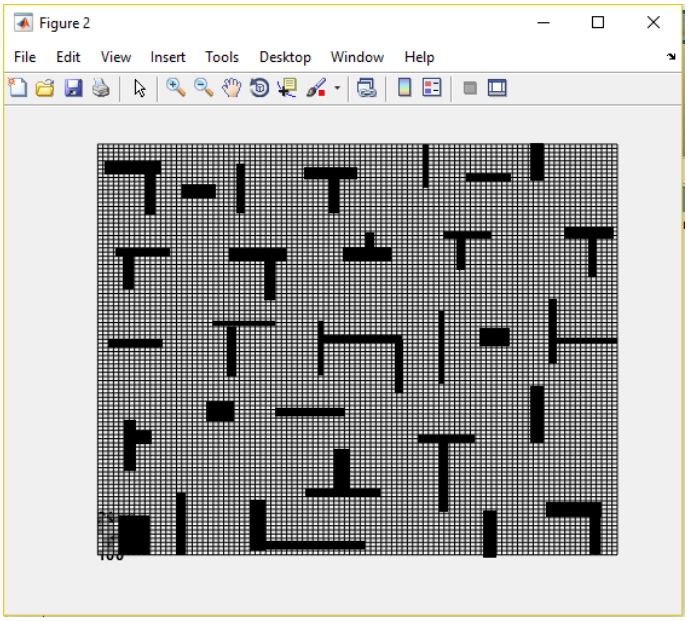

Figure 4.8: Map2: 100*100 Grid environment.

The Figures 4.7 and 4.8 represent the 100*100 grid maps.

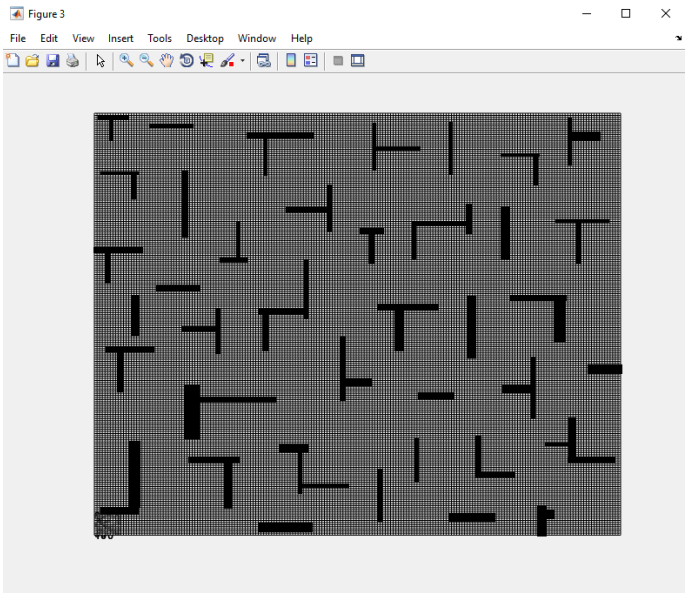

Figure 4.9: Map1: 200*200 Grid environment.

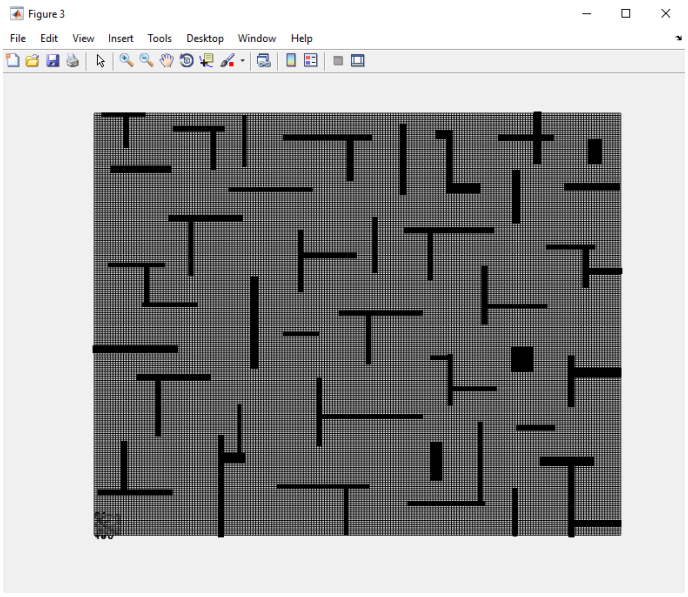

Figure 4.10: Map2: 200*200 Grid environment.

The Figures 4.9 and 4.10 represent the 200*200 grid maps.

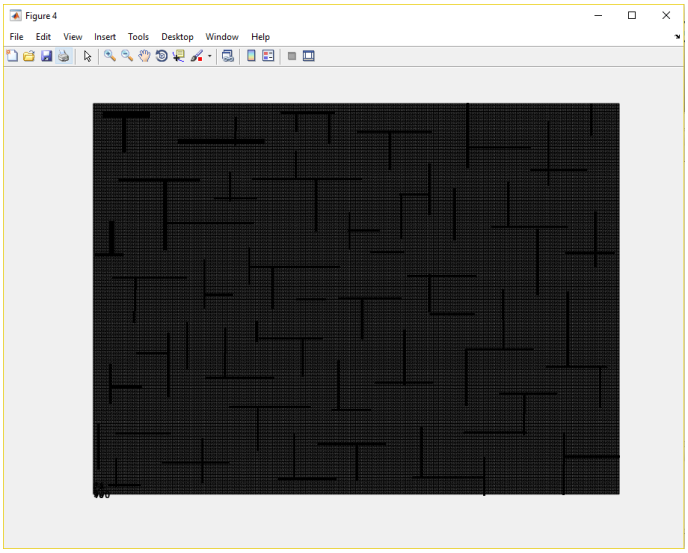

Figure 4.11: Map1: 400*400 Grid environment.

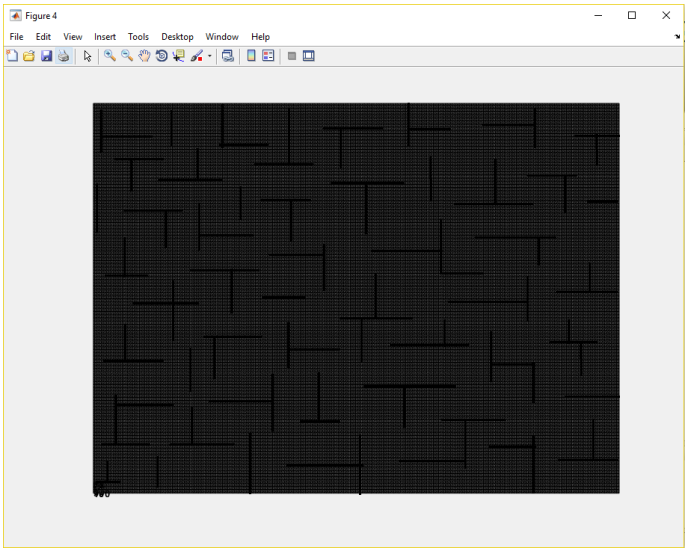

Figure 4.12: Map2: 400*400 Grid environment.

The Figures 4.11 and 4.12 represent the 400*400 grid maps.

In our maps the obstacles are randomly generated. After every iteration (one cycle of the algorithm) a new obstacle is added or the position of the existing obstacles changes or both. We decide the position of the obstacles in the map as we assign each cell that has to be blocked(which means an obstacle is placed in the cells that are blocked). For example, in a map1 we have obstacles in grid g_3, g_6, g_7 . We implement

the algorithm on a map M; where M represents all the scenarios of the maps in which the algorithm was implemented which are denoted as map1, map2, map3 etc. During the first iteration the obstacles are placed in the grid g_3, g_6, g_7 which we can consider as map1; once a best path is found the positions of the obstacles are changed from g_3, g_6, g_7 to g_5, g_8, g_9 which is considered as the map2 on which the algorithm is implemented again with a record of the pheromone values that were updated after finding the best path in map1. Again in map3 the positions of the obstacles are changed and the implemented again without re-initializing it. Similarly, the positions of the obstacles is changed for every iteration and the algorithm is implemented until it reaches the maximum number of iterations. Sometimes, once the position of the obstacles is changed and the obstacle might not be in the grids that are a part of the best path, therefore it would not cause any changes to the path length. As we use the pheromone update rules, by the time the algorithm reaches the maximum iteration the best paths found at every iteration would contain a pheromone value. But there would be one path with the most concentration of pheromone, it is known as the near-optimal path. The length of the near-optimal path is the value of the convergence rate of the algorithm. Therefore, we note the iteration at which the algorithm has begun to converge and gives the same path length(which is the optimal path length) until the maximum iteration is reached. In our approach since we utilize all the iterations as well as all the pheromone values in order to narrow down to the optimal path value; we can say that we do not waste any of the resources or the parameters used.

4.3.2 Collection of the Results

In this subsection we represent the grid maps with the paths obtained after the algorithms were implemented.

The Figures 4.13, 4.14, 4.15, 4.16 represent the 10*10 grids; The Figures 4.17, 4.18,

4.19, 4.20 represent the 20*20 grid maps; The Figures 4.21, 4.22, 4.23, 4.24 represent the 40*40 grid maps; The Figures 4.25, 4.26, 4.27, 4.28 represent the 100*100 grid maps; The Figures 4.29, 4.30, 4.31, 4.32 represent the 200*200 grid maps; The Figures 4.33, 4.34, 4.35, 4.36 represent the 400*400 grid maps. In a 10*10 grid map, the grids are numbered from 1 to 100, with the $grid_1$ being the starting point and the $grid_{100}$ being the destination point. The start and the destination grids are marked with yellow dots in the maps. The blue line represents the predicted path (for the MMAS algorithm) and the red line represents the near optimal path found by the algorithms.

The values obtained (the length of the paths and the other parameters) by implementing these algorithms are shown and discussed in the next section.

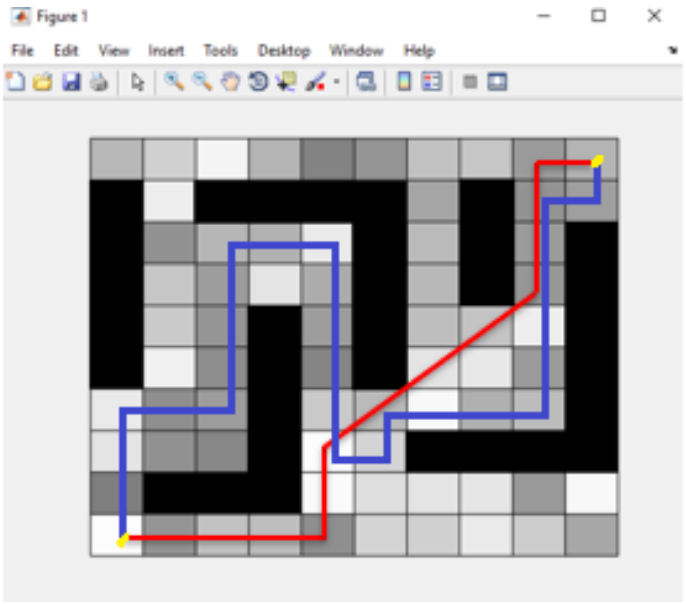

Figure 4.13: Map1: 10*10 Grid environment after the (near) optimal path is found by MMAS.

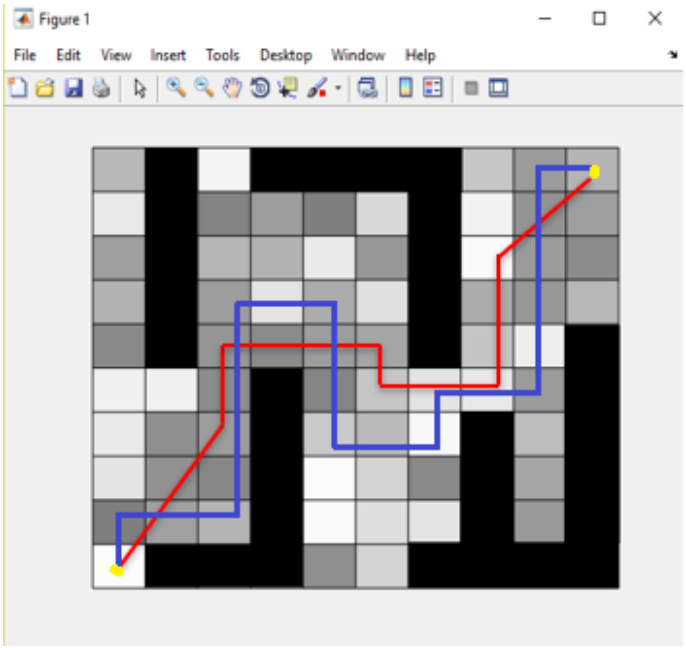

Figure 4.14: Map2: 10*10 Grid environment after the (near) optimal path is found by MMAS.

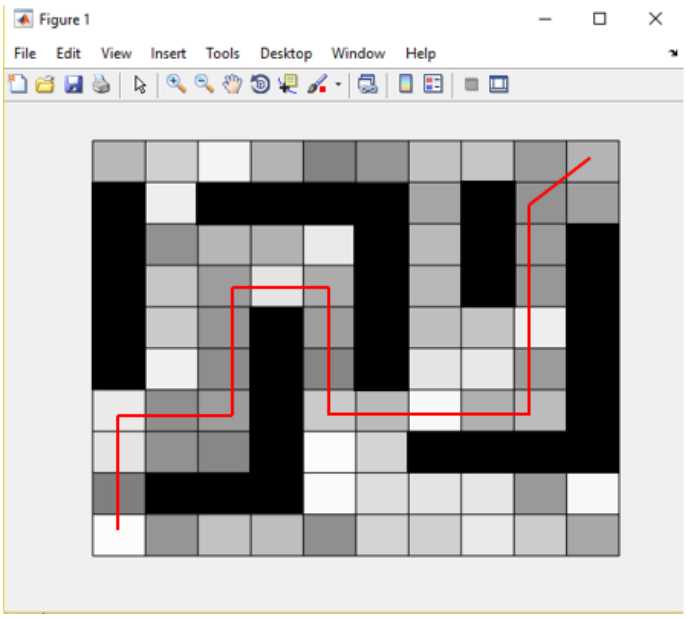

Figure 4.15: Map1: 10*10 Grid environment after the (near) optimal path is found by ACO.

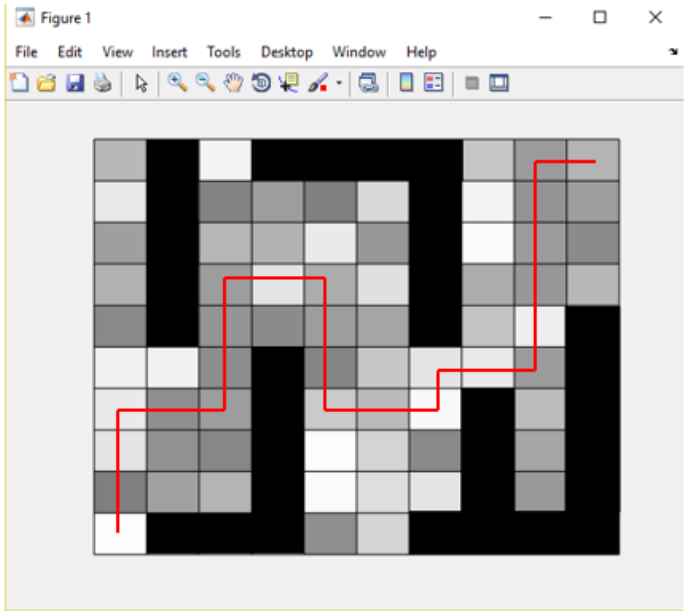

Figure 4.16: Map2: 10*10 Grid environment after the (near) optimal path is found by MMAS.

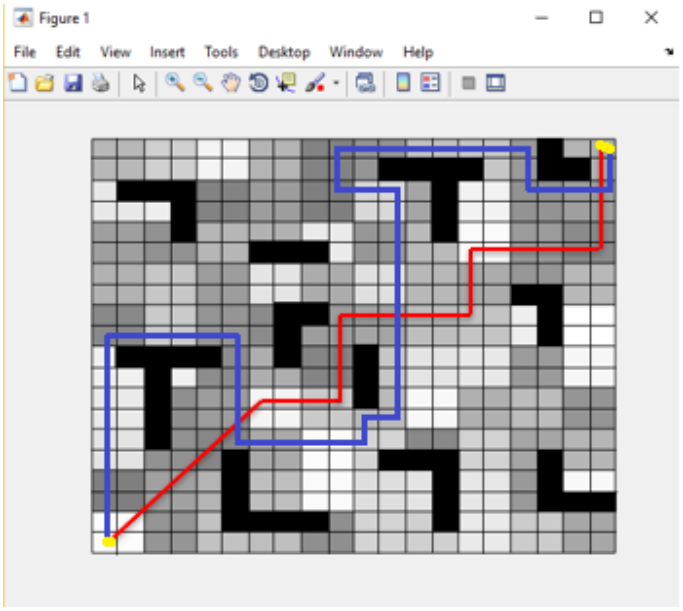

Figure 4.17: Map1: 20*20 Grid environment after the (near) optimal path is found by MMAS.

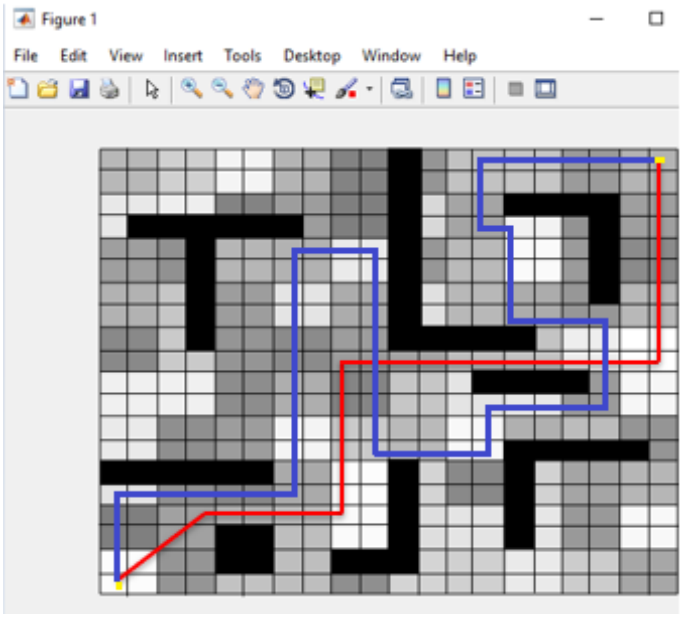

Figure 4.18: Map2: 20*20 Grid environment after the (near) optimal path is found MMAS.

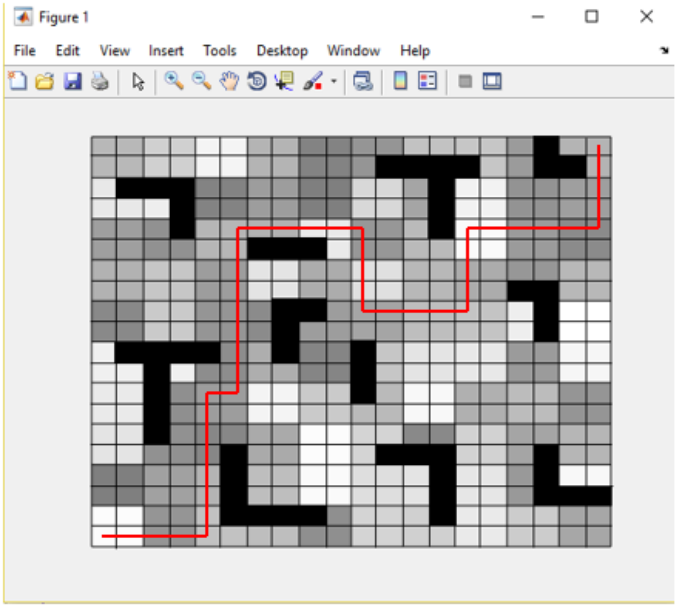

Figure 4.19: Map1: 20*20 Grid environment after the (near) optimal path is found ACO.

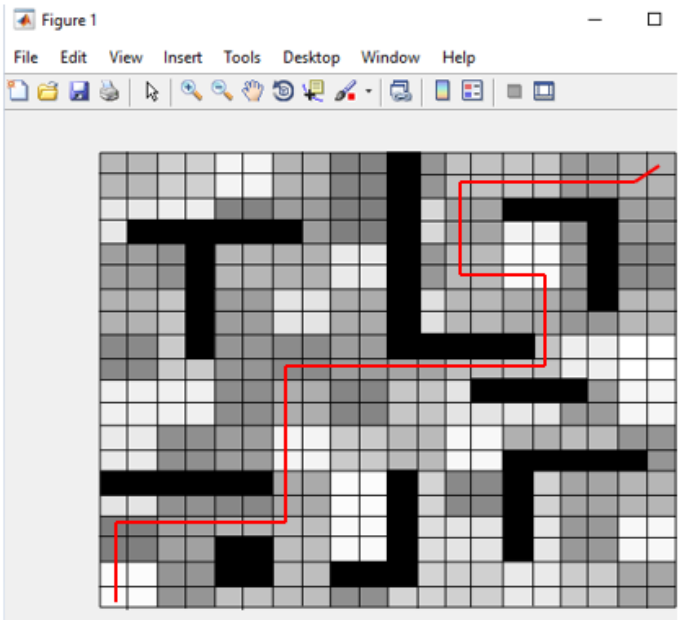

Figure 4.20: Map2: 20*20 Grid environment after the (near) optimal path is found ACO.

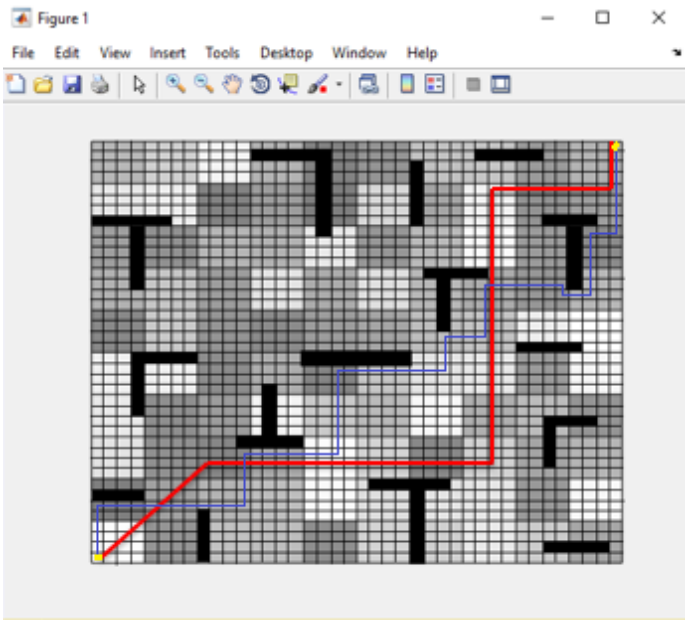

Figure 4.21: Map1: 40*40 Grid environment after the (near) optimal path is found by MMAS.

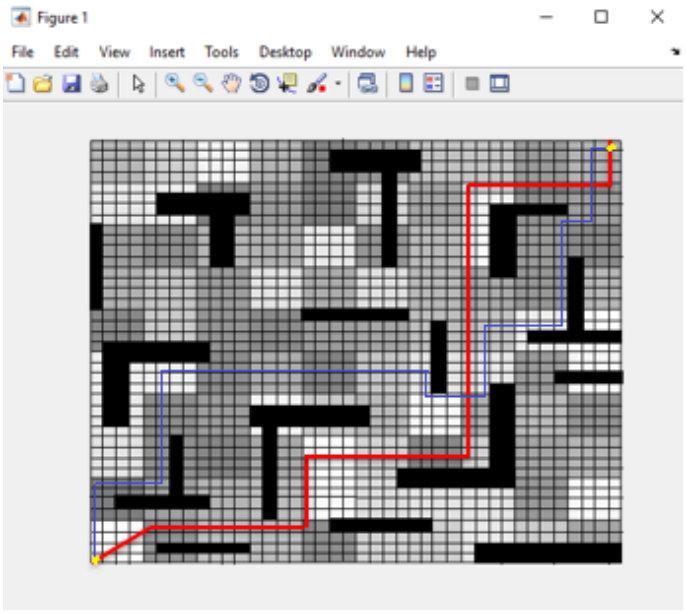

Figure 4.22: Map2: 40*40 Grid environment after the (near) optimal path is found by MMAS.

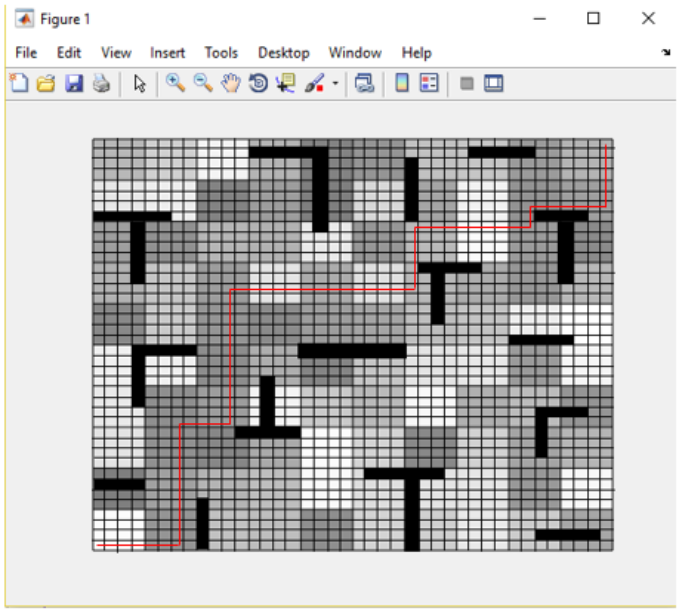

Figure 4.23: Map1: 40*40 Grid environment after the (near) optimal path is found ACO.

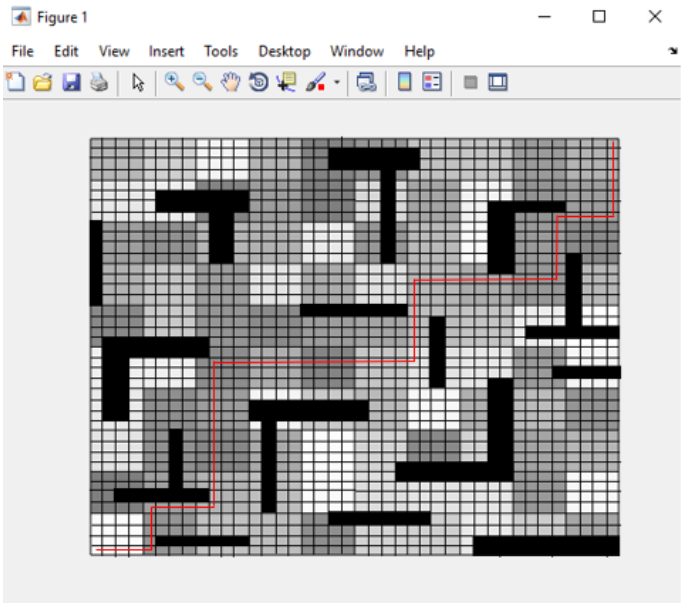

Figure 4.24: Map2: 40*40 Grid environment after the (near) optimal path is found ACO.

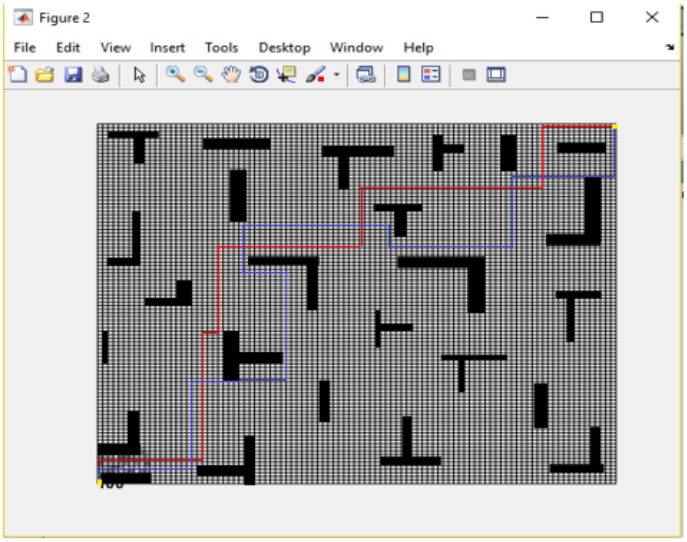

Figure 4.25: Map1: 100*100 Grid environment after the (near) optimal path is found by MMAS.

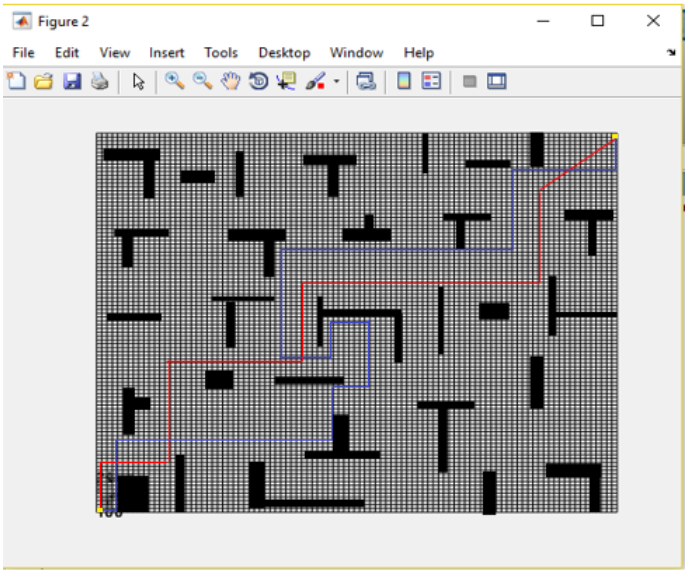

Figure 4.26: Map2: 100*100 Grid environment after the (near) optimal path is found by MMAS.

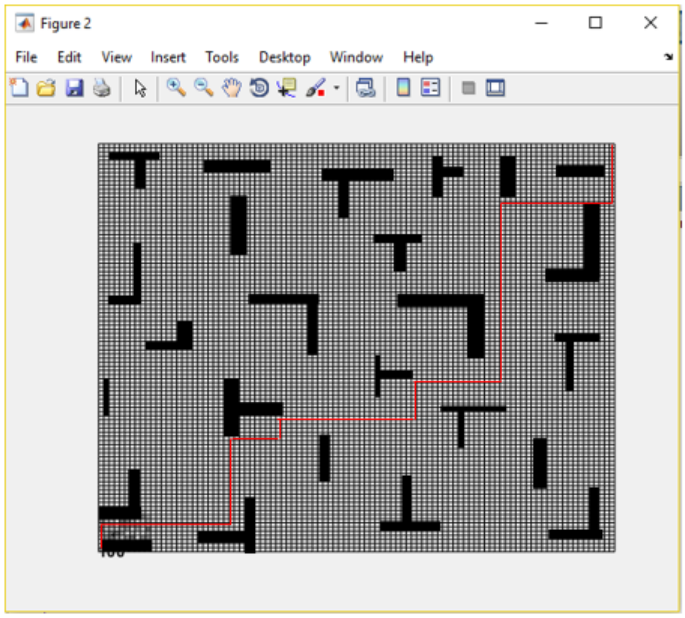

Figure 4.27: Map1: 100*100 Grid environment after the (near) optimal path is found by ACO.

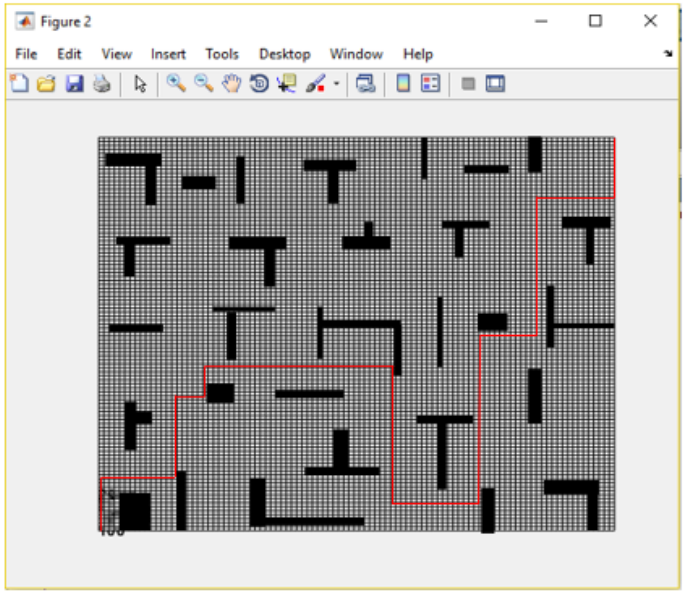

Figure 4.28: Map2: 100*100 Grid environment after the (near) optimal path is found by ACO.

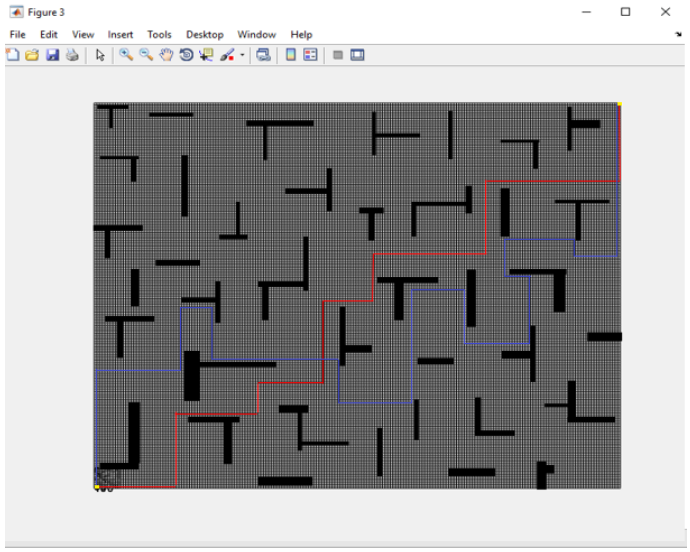

Figure 4.29: Map1: 200*200 Grid environment after the (near) optimal path is found by MMAS.

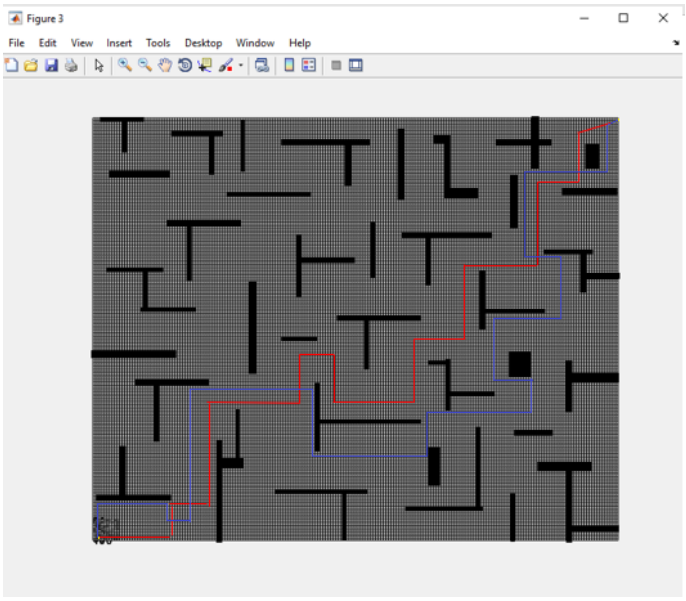

Figure 4.30: Map2: 200*200 Grid environment after the (near) optimal path is found by MMAS.

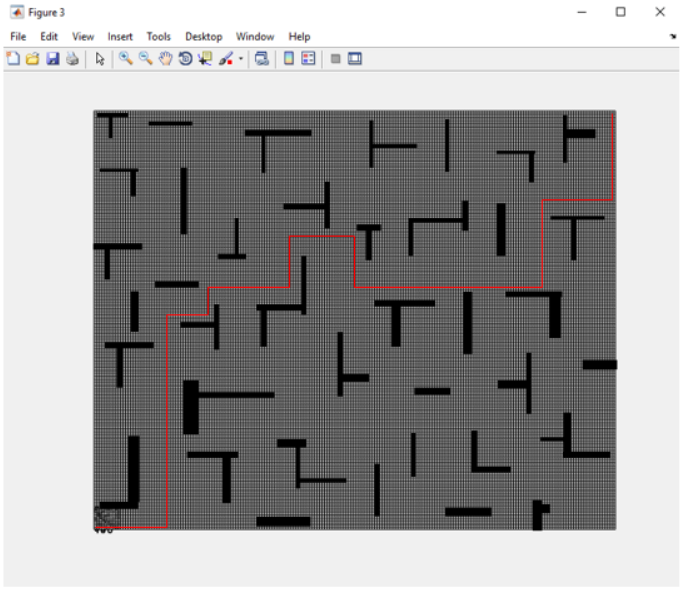

Figure 4.31: Map1: 200*200 Grid environment after the (near) optimal path is found by ACO.

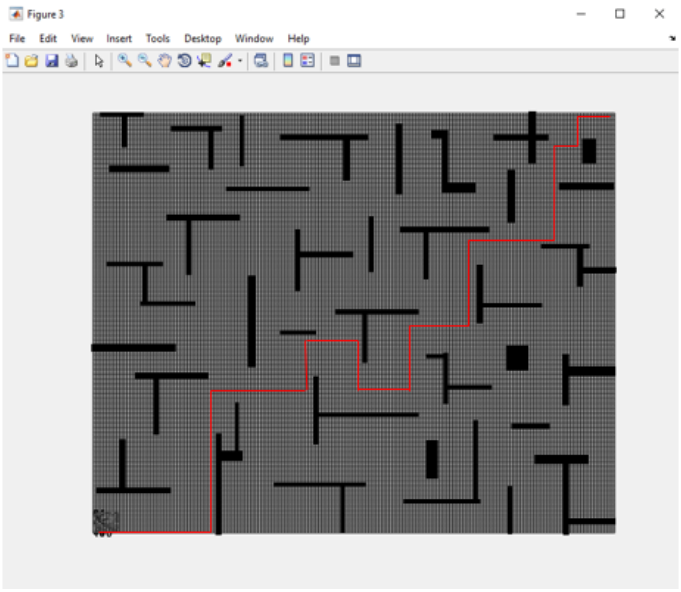

Figure 4.32: Map2: 00*200 Grid environment after the (near) optimal path is found by ACO.

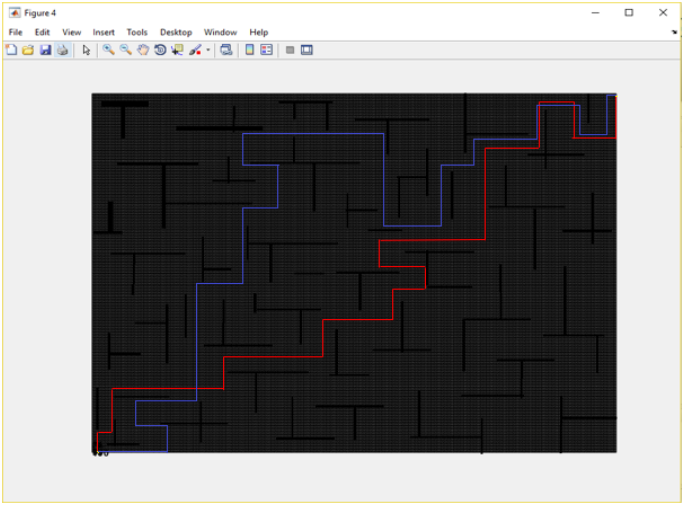

Figure 4.33: Map1: 400*400 Grid environment after the (near) optimal path is found by MMAS.

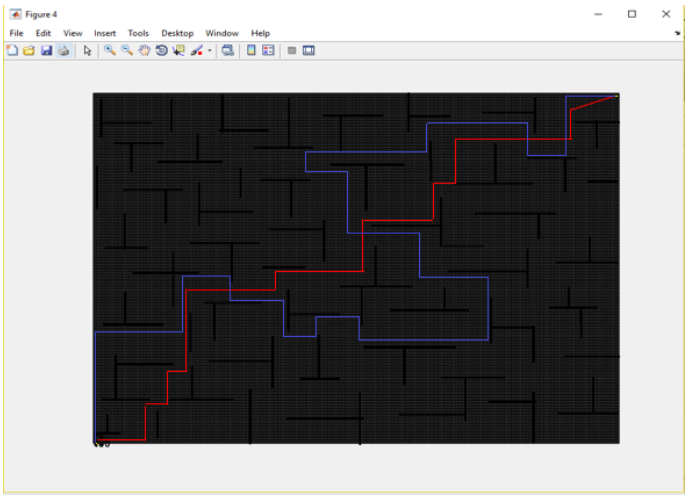

Figure 4.34: Map2: 400*400 Grid environment after the (near) optimal path is found by MMAS.

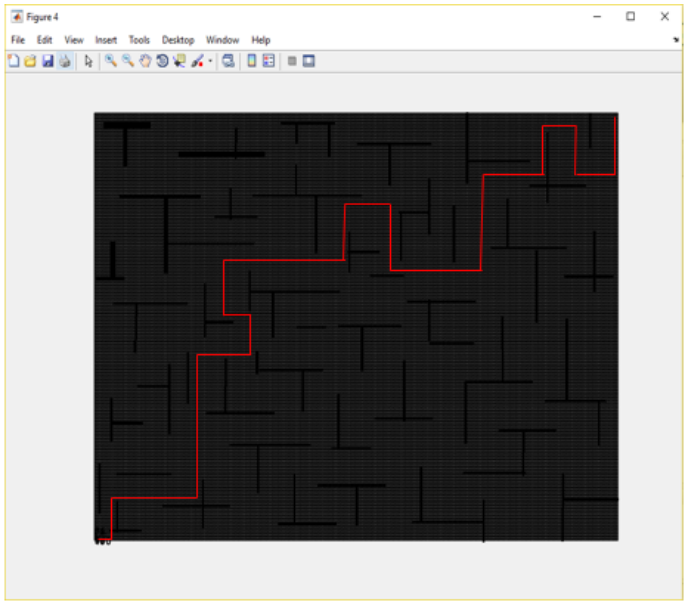

Figure 4.35: Map1: 400*400 Grid environment after the (near) optimal path is found by ACO.

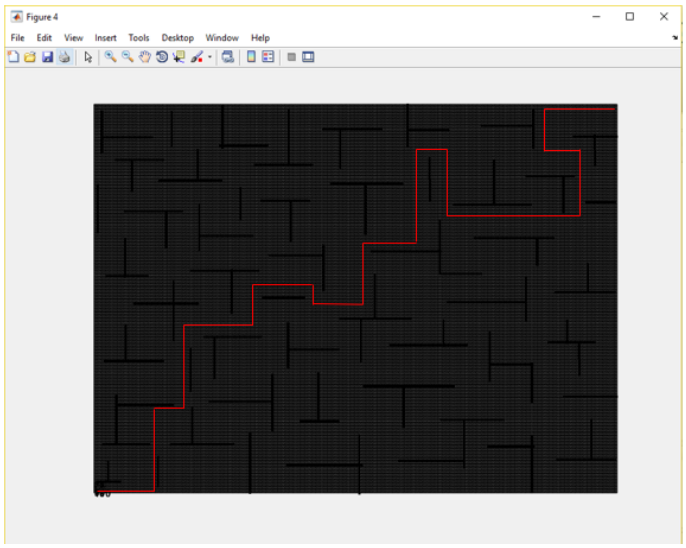

Figure 4.36: Map2: 400*400 Grid environment after the (near) optimal path is found by ACO.

4.3.3 Summary of the values obtained upon implementation of the algorithms

In this section we show and explain the summary of the path length values, time frame and the iteration values obtained after implementing the MMAS and the ACO algorithms.

	Maps	MMAS		Iteration for MMAS		ACO		Iteration for ACO	
		Path Length	Time(seconds)	Optimal	Conv	Path Length	Time(seconds)	Optimal	Conv
Experiment 1	Map1	15.6476	10.35	6	54	24.6476	14.72	9	59
	Map2	17.8955	13.05	8	56	24.8955	16.46	12	62
Experiment 2	Map1	33.7568	16.78	25	216	47.7568	23.45	33	236
	Map2	36.8429	18.56	31	224	47.8439	25.24	44	248
Experiment 3	Map1	84.7652	149.45	96	864	97.5624	153.61	130	944
	Map2	88.6476	168.89	128	896	102.6332	174.43	176	992
Experiment 4	Map1	212.5926	165.79	623	5402	218.6476	174.64	927	5920
	Map2	224.2841	182.96	801	5780	231.4457	210.56	1253	6271
Experiment 5	Map1	492.53	208.91	2405	21601	510.75	217.72	3618	23690
	Map2	523.86	227.59	3244	22495	547.56	241.95	4877	24802
Experiment 6	Map1	1074.56	546.24	9601	86404	1102.62	589.68	14421	94400
	Map2	1124.47	696.79	12804	89609	1149.85	748.46	19220	99204

Table 4.1: Summary of the values obtained after the implementation of MMAS and ACO algorithms.

The Table 4.1 shows the tabulated summary of all the values obtained upon implementing the algorithms (in the columns Iteration for MMAS and ACO; the sub column "optimal" means that the iteration at which the optimal path was found and "conv" means that the iteration at which the algorithm converged to the final solution). From the Table 4.1 we can infer that for the Experiment 1 which consists of the 10*10 grid maps, the near optimal path length obtained by the MMAS algorithm is 15.6476 in 10.35seconds. The optimal path length by the MMAS for this experiment is initially found at the iteration number 6 but later as the algorithm was still building solutions and the path convergence was obtained at iteration number 54. However, the optimal path length obtained by the ACO on the same map is 24.6476 in 14.72 seconds. This optimal path was found by the ACO, initially at iteration number 9 but the path convergence was achieved at iteration number 59. For the next scenario that is the MAP 2 the algorithms are implemented on the same sized grid map where a new obstacle is introduced. In this scenario the MMAS achieves a path length of 17.8955 in 13.05 seconds in the 8th iteration and the convergence rate is obtained at the 56th iteration. On the contrary, the ACO algorithm has achieved a path length

of 24.8955 in 16.46 seconds in the 12th iteration and the convergence rate is obtained at 62nd iteration.

Upon looking at the Table 4.1 carefully, we can notice that for the experiments performed the path length obtained by the MMAS algorithm in almost all of the scenarios is less when compared to the ACO algorithm. It is also clear that the time taken by the ACO algorithm to find a path is also more when compared to the MMAS algorithm.

However, the performance of the MMAS algorithm slows down as we increase the size of the grid maps. In the Experiment number 2 where we use a 20*20 grid map number 1 the optimal path length obtained is 33.7568 in 16.78 seconds and the path length was obtained at 25th iteration whereas the convergence was achieved at 216th iteration. For the same map the path length obtained by the ACO is 47.7568 in 23.45 seconds at 33rd iteration and converges at the 236th iteration. The values obtained by the algorithms when implemented in a 40*40 grid map that is the Experiment number 4 are 84.7652 in 149.45 seconds at 96th iteration and convergence is at 864th iteration; for the MMAS. In the same map the path length is 97.5624 in 153.61 seconds at 130th iteration and the convergence at the 944th iteration; for the ACO.

A similar kind of increasing trend is observed as we go through each column from the Experiment 1 to Experiment 6 for different parameters such as the path length, time, iteration at which the optimal path was found and the iteration at which the convergence of the solution has occurred for both MMAS and the ACO algorithm. Since the size of the grid is increased that is the search space is increased the processing time for the algorithms also increases. We will further evaluate the results obtained in the Figure **??** in the section 4.4.

4.4 Evaluation of the Results

4.4.1 Performance Evaluation Indexes

In order to comprehensively measure the performance of the ant colony algorithm, the following basic indexes to evaluate the performance of ant colony algorithm are used which have been introduced in [28]:

The Best Performance Index:

Let E_O represents the best performance index, the formula is as follows:

$$E_o = \frac{c_b - c^*}{c^*} * 100 \tag{4.2}$$

where c_b represents the optimal value obtained by the algorithm; c* represents the theoretical optimal value. When the theoretical optimal value is unknown,it is replaced by the best known value. The optimal performance index is used to measure the optimal optimization degree of the ant colony algorithm. The smaller of the value means that the optimal performance of the ant colony algorithm is better.

The Time Performance Index

Let E_T represents the time performance index, the formula is as follows:

$$E_T = \frac{I_a * T_0}{I_{max}} * 100 \tag{4.3}$$

where I_a represents the algorithm's number of iterations when it meets convergence condition(In this paper, Ia refers to the iteration number when the mean path length tend to be stable); T_0 represents the average execution time of one iteration; I_{max} represents the algorithm's number of iterations. The time performance index is used to measure the search speed of the ant colony algorithm. The smaller of the E_T

means that the convergence speed of the ant colony algorithm is quicker.

The Robustness Performance Index

Let E_R represents the robustness performance index, the formula is as follows:

$$E_R = \frac{c_a - c^*}{c^*} * 100 \tag{4.4}$$

where c_a represents the average path length value; c* represents the theoretical optimal value. When the theoretical optimal value is unknown, it is replaced by the best known value. The smaller of the E_R means that the stability of the ant colony algorithm is better.

4.4.2 Performance Evaluation of the Algorithms

The performance evaluation results of the algorithms implemented are given in the Table 4.2.

		MAP1			MAP2		
		E_o	E_T	E_R	E_o	E_T	E_R
Experiment 1	MMAS	0.18	2.26	0.6	0.25	2.34	1.1
	ACO	0.27	3.62	1.0	1.8	4.88	2.7
Experiment 2	MMAS	0.39	2.75	2.6	0.43	3.18	2.51
	ACO	1.46	3.4	4.72	2.91	5.49	5.16
Experiment 3	MMAS	0.85	2.92	8.6	1.3	3.98	4.83
	ACO	3	4.11	9.1	4.16	6.78	11.2
Experiment 4	MMAS	1.2	3.28	12.4	1.8	4.26	14.85
	ACO	4.1	4.7	16.75	5.66	7.9	19.1
Experiment 5	MMAS	1.5	5.64	14.93	2.84	5.82	17.9
	ACO	4.6	8.13	19.28	7.2	9.16	24.18
Experiment 6	MMAS	2	9	18.52	3.8	7.2	21.2
	ACO	5.3	13.2	22.31	9.74	11.35	29.37

Table 4.2: Performance Evaluation of the results.

Looking at the performance evaluation results shown above in the Table 4.2, we can incur that the Best performance index of the MMAS algorithm is always less than that of the ACO algorithm. For example, in Experiment 1; for the MMAS and the ACO the $E_0 = 0.18 percent$ and $E_0 = 0.27 percent$. It means that the performance of the MMAS algorithm was better in the MAP 1 when compared to the performance of the ACO algorithm. The time performance index and the robustnes performance index are also evaluated for. A similiar kind of performance trend as in MAP 1 was observed when the algorithms' performances were evaluated in MAP 2.

To study an compare the performances of the algorithms in the experiments we would have to keep looking row-wise in Table 4.2 to understand the comparision between the two algorithms.

As mentioned in the Subsection 4.4, the smaller the values of E_0, E_T, E_R; the better is the performance of the algorithm. Therefore, if we compare all the values of the MMAS performance indexes to the values of the ACO performance indexes we can say that the MMAS has performed better than the ACO algorithm in the experiments.

The Figure 4.37 and 4.38 are the graphical representation of the comparison of the performance evaluation indexes of the two algorithms in the MAP 1 and MAP 2; as shown in the Table 4.2.

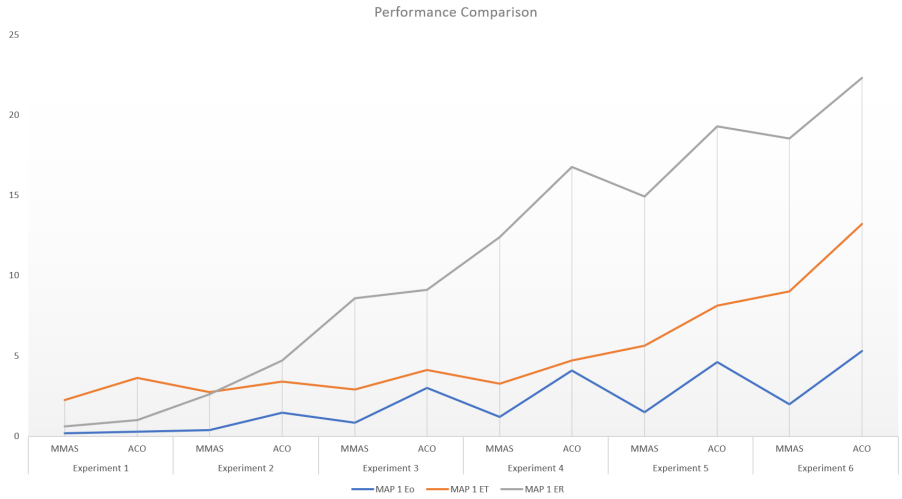

Figure 4.37: Performance comparison of MMAS and ACO in MAP 1.

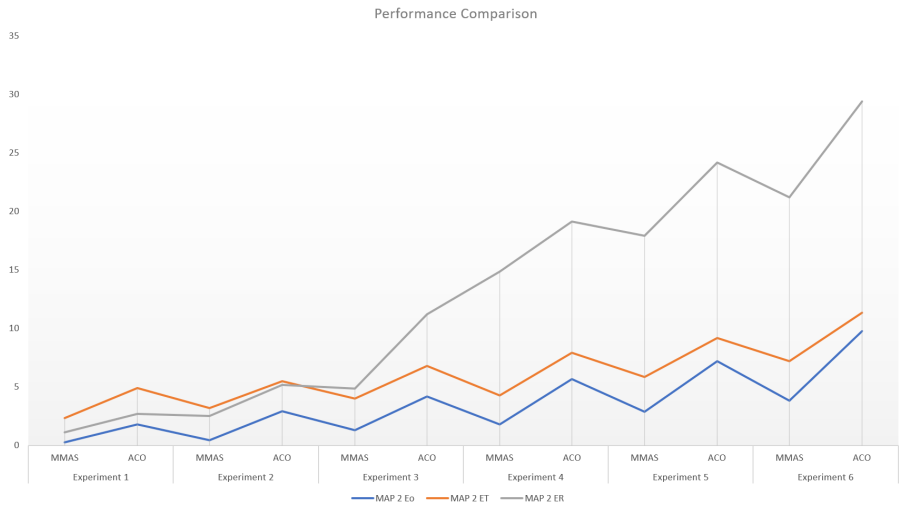

Figure 4.38: Performance comparison of MMAS and ACO in MAP 2.

4.5 Discussion

In the Experiment 1 implemented on 10*10 grid maps, the MMAS and the ACO algorithms have performed well because they were able to converge to a near optimal solution in the end. However, the MMAS was able to converge to a solution faster.

In the Experiment 2 and 3 where the MMAS and the ACO Algorithms were implemented on a 20*20 and 40*40 grid maps, the MMAS algorithm was still able to find the solutions quicker than the ACO algorithm.

However in Experiment 4,5 and 6(in the 100*100, 200*200, 400*400); the MMAS has slowed down a bit when compared to its speed in the previous Experiments 1,2,3. The ACO algorithm has been performing relatively slow when compared to the MMAS overall. As the size of the search space that is the size of the maps increases the convergence speed of the algorithms has reduced drastically.

On comparing the performance of this algorithm with that of the ACO on all the six set of map sizes, we can say that the performance of the MMAS is better than that of the ACO algorithm because the ACO converges to a solution late than the improved MMAS algorithm. Larger grid maps such as 40*40, 100*100, 200*200 and 400*400 haven't been used before by researchers to implement the algorithm and check for its scalability. Therefore experiments have been conducted in the larger grid maps which are also one of the main contributions of this thesis. The optimal path obtained by the algorithm in the environment after the change in the obstacles position is shown in the Figures 4.13, 4.14, 4.3, 4.4, 4.21, 4.22, 4.25, 4.26, 4.29, 4.30, 4.33, 4.34 . Comparing the time taken by the algorithm we can see that its capability to do so is more in the smaller environments even if the agents had to reroute their path because of the increase in the workspace area.

In the grid environment the MMAS algorithm has a range of the pheromone values that can be deposited by the ants while traversing through the environment.According to the theory of the addition of diversification mechanism as stated in 2.4.1 and 2.4.2,

the best solution that has been found since the re-initialization of the pheromones is used which is in turn increasing the search diversification. That helped the MMAS algorithm to converge to a better quality solution which is the near optimal path in our experiments. Also, using the technique of adding local search routines where the algorithm is constantly searching for better solution in its neighborhood and if one such solution is found then it replaces that with the current solution. The algorithm has repeated this process until no better solution could be attained in the neighborhood.

Therefore, We could anticipate that the smaller of the grid size, the more accurate the representation of the obstacles. But at the same time, a considerable amount of storage is required, and the search range of the algorithm will increase exponentially.

Chapter 5

Conclusion and Future Work

This thesis mainly focuses on the mobile Robot Path Planning Problems based on the Ant Colony Optimization variant algorithm, i.e., the Max-Min Ant System Algorithm. To be specific, the research work of the thesis is listed below:

- Firstly, the actual working environment of the mobile robot is modeled. Environmental modeling adopts grid method. The actual working environment is divided into grids of the same size, and the grids are containing obstacles are grayed out. Two methods; Serial number method and the rectangular coordinate to identify all grids and they could be well commuted to each other. The other Robot Path Planning problem based on the Ant Colony Algorithm is a process that through the interaction and cooperation between the individual of the ant colony, they will avoid all obstacles to find an optimal path from the starting grid to the target grid.

- Secondly, we made modifications to the MMAS algorithms as per the techniques introduced by previous researchers . For example, we applied diversification mechanisms based on the pheromone initialization and addition of local search routines that are described in Section 2.4

- Lastly, Based on the MATLAB platform the thesis tests the algorithm on grid

maps of 6 different sizes to verify the validity and effectiveness of the MMAS Algorithm under the different maps used.

Several conclusions could be summarized through the results of the six group experiments:

- The ability of the MMAS algorithm to find the optimal solution, convergence speed and stability are higher when implemented on a grid map of smaller size.
- When dealing with the grid maps of larger sizes the ability of the algorithm to search the near optimal solution is same but the time taken is on an increasing trend with the increase in the size of the map.
- However, the speed and stability are fast in a smaller environment when compared to the large maps, no matter how the complexity of the map and the start and the destination path vary.

We could conclude by saying that MMAS is suitable and effective in solving RPP considering its processing time and stability. However, the research work in this thesis also has some limitations mainly reflecting on the following aspects.

- The experiments in this thesis are set on a grid map, instead performing on the real-time environment could be done in the future work.
- However, since we mentioned that the larger the grid size the obstacles couldn't be represented accurately and the time taken to find the solution is more. So an attempt to reduce the amount of time taken to find a solution on the grid maps of the size larger than 400*400 could be an extended work of this thesis.

Bibliography

[1] Jens Christian Andersen. Mobile robot navigation. 2007.

[2] Christian Blum. Ant colony optimization: Introduction and recent trends. *Physics of Life reviews*, 2(4):353–373, 2005.

[3] Johann Borenstein and Yoram Koren. Optimal path algorithms for autonomous vehicles. In *Proceedings of the 18th CIRP Manufacturing Systems Seminar*, pages 5–6, 1986.

[4] Michael Brand, Michael Masuda, Nicole Wehner, and Xiao-Hua Yu. Ant colony optimization algorithm for robot path planning. In *Computer Design and Applications (ICCDA), 2010 International Conference on*, volume 3, pages V3–436. IEEE, 2010.

[5] N Buniyamin, N Sariff, WAJ Wan Ngah, and Z Mohamad. Robot global path planning overview and a variation of ant colony system algorithm. *International journal of mathematics and computers in simulation*, 5(1):9–16, 2011.

[6] Marco Dorigo. Optimization, learning and natural algorithms. *PhD Thesis, Politecnico di Milano*, 1992.

[7] Marco Dorigo, Mauro Birattari, Christian Blum, Maurice Clerc, Thomas Stützle, and Alan Winfield. *Ant Colony Optimization and Swarm Intelligence: 6th In-*

ternational Conference, ANTS 2008, Brussels, Belgium, September 22-24, 2008, Proceedings, volume 5217. Springer, 2008.

[8] Marco Dorigo and Luca Maria Gambardella. Ant colony system: a cooperative learning approach to the traveling salesman problem. *IEEE Transactions on evolutionary computation*, 1(1):53–66, 1997.

[9] Marco Dorigo, V Maniezzo, and A Colorni. Positive feedback as a search strategy. dipartimento di elettronica, politecnico di milano. Technical report, Italy, Tech. Rep. 91-016, 1991.

[10] Marco Dorigo, Vittorio Maniezzo, and Alberto Colorni. Ant system: optimization by a colony of cooperating agents. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, 26(1):29–41, 1996.

[11] Marco Dorigo and Thomas Stützle. The ant colony optimization metaheuristic: Algorithms, applications, and advances. In *Handbook of metaheuristics*, pages 250–285. Springer, 2003.

[12] Hai-Bin Duan. Ant colony algorithms: theory and applications. *Chinese Science*, 2005.

[13] Luca Maria Gambardella and Marco Dorigo. Solving symmetric and asymmetric tsps by ant colonies. In *Evolutionary Computation, 1996., Proceedings of IEEE International Conference on*, pages 622–627. IEEE, 1996.

[14] Gengqian Liu, Tiejun Li, Yuqing Peng, and Xiangdan Hou. The ant algorithm for solving robot path planning problem. In *null*, pages 25–27. IEEE, 2005.

[15] Mauro Birattari Marco Dorigo and Thomas Stutzle. Sensor fusion in certainty grids for mobile robots.

[16] Mauro Birattari Marco Dorigo and Thomas Stutzle. Ant colony optimization artificial ants as a computational intelligence technique. *IEEE COMPUTATIONAL INTELLIGENCE MAGAZINE*, pages 28–39, 2006.

[17] Hui Miao and Yu-Chu Tian. Robot path planning in dynamic environments using a simulated annealing based approach. 2008.

[18] Adam Milstein. Occupancy grid maps for localization and mapping. In *Motion Planning*. InTech, 2008.

[19] G Nirmala, S Geetha, and S Selvakumar. Mobile robot localization and navigation in artificial intelligence: Survey. *Computational Methods in Social Sciences*, 4(2):12, 2016.

[20] Purushothaman Raja and Sivagurunathan Pugazhenthi. Optimal path planning of mobile robots: A review. *International Journal of Physical Sciences*, 7(9):1314–1320, 2012.

[21] Valéria de C Santos, Fernando S Osório, Cláudio FM Toledo, Fernando EB Otero, and Colin G Johnson. Exploratory path planning using the max-min ant system algorithm. In *Evolutionary Computation (CEC), 2016 IEEE Congress on*, pages 4229–4235. IEEE, 2016.

[22] Alexander Schrijver. *A course in combinatorial optimization*. TU Delft, 2000.

[23] Thomas Stützle and Marco Dorigo. Aco algorithms for the traveling salesman problem. *Evolutionary algorithms in engineering and computer science*, pages 163–183, 1999.

[24] Thomas Stützle and Holger H Hoos. Max–min ant system. *Future generation computer systems*, 16(8):889–914, 2000.

[25] Ioan Susnea, Viorel Minzu, and Grigore Vasiliu. Simple, real-time obstacle avoidance algorithm for mobile robots. In *8th WSEAS International Conference on Computational Intelligence, Man-Machine Systems and Cybernetics (CIMMACS09)*, 2009.

[26] Guan-Zheng Tan, Huan He, and Aaron Sloman. Ant colony system algorithm for real-time globally optimal path planning of mobile robots. *Acta automatica sinica*, 33(3):279–285, 2007.

[27] Sebastian Thrun, Wolfram Burgard, and Dieter Fox. *Probabilistic robotics*. MIT press, 2005.

[28] Chenhan Wang. Comparative research on robot path planning based on ga-aca and aca-ga. 2017.

Appendix-Abbreviations

- ACO : Ant Colony Optimization
- ACS : Ant Colony System
- AS : Ant System
- MMAS: Max-Min Ant System
- CO : Combinatorial Optimization
- TSP : Travelling Salesman Problem

Vita Auctoris

NAME:	Satya Shree Sankini
PLACE OF BIRTH:	Telangana,India
EDUCATION:	Bachelor of Technology in Computer Science, GuruNanak Institutions Technical Campus, Telangana, India, 2016. Master of Science in Computer Science, University of Windsor, Windsor, Ontario, Canada, 2019.