UNIVERSITY OF CALIFORNIA
RIVERSIDE

Distributed Multi-Robot Collaboration Using Evolutionary Computation

A Thesis submitted in partial satisfaction
of the requirements for the degree of

Master of Science

in

Computer Science

by

Amirali Darvishzadeh

December 2011

Thesis Committee:

Professor Bir Bhanu, Chairperson
Professor Christian Shelton
Professor Anastasios Mourikis

The Thesis of Amirali Darvishzadeh is approved:

Committee Chairperson

University of California, Riverside

Acknowledgments

I would like to show my gratitude to Dr. Bir Bhanu for his support on this work. I also wish to thank Negin E. who have helped me in organizing my thesis. Thanks are also due to Angelo P. to help me set up multi-robot system.

To the Green people of Iran.

ABSTRACT OF THE THESIS

Distributed Multi-Robot Collaboration Using Evolutionary Computation

by

Amirali Darvishzadeh

Master of Science, Graduate Program in Computer Science
University of California, Riverside, December 2011
Professor Bir Bhanu, Chairperson

Robots are utilized in activities such as mine detection, search and rescue, etc. Especially where the environment is dangerous or search time can be optimized, etc. As the cost of robots reduced and application of robotic systems grew, researchers started working on distributed robotic systems. However, creating a robust, flexible and scalable system of collaborating mobile robots that can efficiently perform the search task has been a challenging issue for roboticists. Biologists observed that animal interactions in their societies create a collective behavior that helps them in foraging, group defending and other activities. They show that each animal in its society has a simple role but the collection of these behaviors enables them to accomplish a complex task. Moreover, swarm behavior has interesting properties such as robustness, flexibility and scalability. Based on these studies, researchers have introduced new optimization methods. Roboticists utilize these optimization methods to provide important properties of swarm behavior. PSO is one of the successful optimization methods in this area. Initially, PSO performs well in exploring different search regions, however, in some cases the method doesnt exploit well in promising

regions which increases the search time. In this study, mobile robots are deployed to find a target in the search space. Robots interact with each other and move around the search space using PSO algorithm. In addition, the MPSO algorithm is introduced to create an efficient balance between exploration and exploitation of the PSO algorithm. Results show that MPSO reduces search time and minimizes region revisiting.

CONTENTS

LIST OF FIGURES

1 Introduction

Robots are utilized in activities such as mine detection, search and rescue, etc. Especially where the environment is dangerous or search time can be optimized, etc. Early research in robotics area mostly focused on single-robot systems. As the cost of robots reduced and application of robotic systems grew, researchers started working on distributed robotic systems. However, creating a robust, flexible and scalable system of collaborating mobile robots that can efficiently perform the search task has been a challenging issue for roboticists.

By increasing the number of robots in a system, control of the system becomes unreliable and sometimes infeasible. To overcome this problem, new approaches are developed to distribute control methods. In the mid 1980 Brooks [6] introduced the concept of behavioral robotics. In this method each individual in the system interact with others to create desired emergent behavior. Khatib presented another approach in [8] that is called artificial potential. Method was initially introduced to be utilized in obstacle avoidance for mobile robot systems and also it has been applied to other fields such as autonomous robot path planning. The idea is to create a workspace for multi-robot systems that each individual approaches the goal state while avoiding potential collisions.

Some of the studies by biologists captured the attention of the roboticists. Biologists observed that animal interactions in their societies create a collective behavior that helps them in foraging, group defending and other activities. They show that each animal in its society has a simple role but the collection of these behaviors enables them to accomplish a complex task. Moreover, swarm behavior has interesting properties such as robustness, flexibility and scalability. Based on these studies, researchers have introduced new optimization methods.

Roboticists utilize these optimization methods to provide important properties of swarm behavior. PSO is one of the successful optimization methods in this area and is inspired by

observing fish schooling and bird flocking. The method deploys a swarm of particles to move around the search space by collaborating and changing their velocity direction. Initially, PSO performs well in exploring different search regions, however, in some cases the method doesn't exploit well in promising regions which increases the search time.

In this study, mobile robots are deployed to find a target in the search space. Robots interact with each other and move around the search space using PSO algorithm. In addition, the MPSO algorithm is introduced to create an efficient balance between exploration and exploitation of the PSO algorithm. Results show that MPSO reduces search time and minimizes region revisiting.

1.1 Outline

To test the performance of the optimization methods, a mobile robot system is set up to find a target in a search space. Mobile robot system is composed of three different robots (details of the robots are described in Experimental results section) built by MobileRobots Inc. To simplify the object detection system, target has a unique color in the search spaces. During experiments, either a red pillow or a yellow box is selected as a target and mobile robots collaborate with each other and move around the search space to find the target. Two different search spaces are considered to conduct the experiments; one is a small room and the other one is a courtyard.

Optimization methods that are studied in this thesis include: genetic algorithm and particle swarm optimization. Moreover, to measure and compare the performance of the methods with each other, the robots perform Exhaustive search as a baseline method and an improved version of the particle swarm optimization method is introduced and called MPSO.

Map of the search spaces are created and given to the robots. During the experiments mobile robots are equipped with different sensors such as sonar, infra-red and laser that enable the

robots to perform localization task and locate them on the map. In addition, a communication mechanism is created to let the robots to exchange their understanding about the search space.

To facilitate the search and path planning tasks, search space is discretized into safe and unsafe units and robot only move into safe units. To keep track of visited regions, when a robot visits a unit (center of the unit falls into the field of view of the robot's camera), it would be marked as visited unit and will be removed from search list.

Following aspects of multi-robot collaboration are studied in this thesis:

1) Search time required for each method.
2) Suitable size of the unit to discretize the map is learned.
3) Performance of the search methods is compared.
4) Robustness and scalability of the system are discussed.

1.2 Organization

Contents of the thesis are organized into following chapters: In Chapter 2, we start with swarming behavior in animal societies and its motivations to be applied in multi-robot systems. In addition, the problem is introduced and some related works to handle issues in robotic systems are explained. In Chapter 3, we will present our solution to the problem and will mention some important details of the implementations. Moreover, coordination and motion strategies for each method are explained. In Chapter 4, performance of the methods in different situations is compared with each other. Furthermore, we will study the following important properties of the system: scalability and robustness. In Chapter 5, we conclude and state some future works for our thesis.

2 From swarm behavior to multi-robot coordination

Swarming is an interesting behavior that happens in animal societies. This type of behavior has several properties including robustness, flexibility and scalability. These properties have been challenging issues for roboticists in designing robotic systems. In this section, first, swarm behavior and its properties are discussed, then we will discuss that how we can apply this concept in robotic area.

2.1 Swarm

The word swarm behavior or swarming conveys the behavior in animal societies in which each member of the society performs simple role, but the entire group creates complex behavior. The term swarming is originally applied to insects, but it can also be used for other animal societies, which have swarm behavior. Some of the common terms used for swarm behavior of animals, including flocking, which is specifically used for swarm behavior of birds, herding refers to swarm behavior of quadrupeds, schooling describes swarming in fishes [3]. Figure 2.1 shows some of the swarming examples in animal societies.

(a) (b) (c)

Figure 2.1. Various instances of swarming in animal societies: (a) birds flocking, (b) flock of sheep walking together, (c) fish schooling

Researchers in computer science have adopted swarming idea to deal with complex problems. Swarming is particularly applied to the problems in which achieving a goal by a single agent is impossible, and overall result or performance of the system can be improved if multi-agent system is used instead of single agent.

2.2 Swarm intelligence

Gerardo Beni and Jing Wang introduced swarm intelligence (SI) concept for the first time in [16]. SI is a population-based stochastic method and deals with combinatorial optimization problem. In other words, SI is the aggregation of behavior of decentralized, self-organized systems, either naturally or artificially. Decentralized and self-organized are two important properties in SI system, their description and importance are as following:

1) **Self-organization:** one of the important aspects of swarm behavior is self-organization. It refers to the property of a system where the pattern or structure of the system appears only in the interaction of the entities in the system and not by imposing an external planning to the entities [2]. In other words, organization of the entities in the swarm system comes from swarm system and not outside of it. Researchers have adopted the concept of self-organization, which originally comes from social science field, to create a multi-agent system in which the agents are not explicitly designed to achieve a goal in a cooperated manner, but they operate in a self-interested way.
2) **Decentralized system:** decentralization is the process of spreading decision-making over the entities in the system. There are several advantages in decentralized mobile robot systems compared with centralized ones. Most importantly they make the system more robust with respect to the failure of some robots. Moreover they don't need a control module with high computational capabilities to organize the system.

A SI system consists of a population of simple entities, and inspires its interaction mechanism by observing social interactions of animal societies such as bird flocking or fish schooling. Entities involved in SI systems have two important interactions; a local interaction that refers to interaction of the entity with the environment, and a global interaction with other entities in the system to exchange their understanding of the environment. Communications or interactions between entities with environment or entities with entities take place either in a direct way (such as visual contact) or indirectly (for example invisible scents existing in the environment).

There are two types of SI: swarm-formation and stigmergy. Swarm-formation refers to the creation of swarm of entities that is a basic form of SI. This form of SI can be observed in flocks of birds and shoals of fish, etc. There are two basic rules for this type of SI: stay close to the swarm system, but avoid coming too much close to individuals.

Stigmergy is the other form of SI and is a mechanism for indirect coordination of entities. The terms stigmergy and swarm intelligence are usually used interchangeably in scientific articles. Stigmergy describes swarm behaviors that are indirectly controlled by the environment. For example in ant colonies, ants use pheromones and scents to exchange their understanding of environment with each other. This type of communication is volatile, but it allows ants to create trails for foraging dynamically. Swarm-formation and stigmergy are depicted in figure 2.2.

2.3 Swarm robotic

Camazine *et al.* in [1] investigated the coordination of members in social insects, and they show that there is no centralized coordination mechanism in such a society, but the entire system is robust, scalable and flexible. These properties captured attention of researchers in multi-robot area to achieve those properties by mimicking swarm behavior existing in the

nature. Thus, roboticists started to apply swarming behavior in multi-robot system and called it swarm robotic.

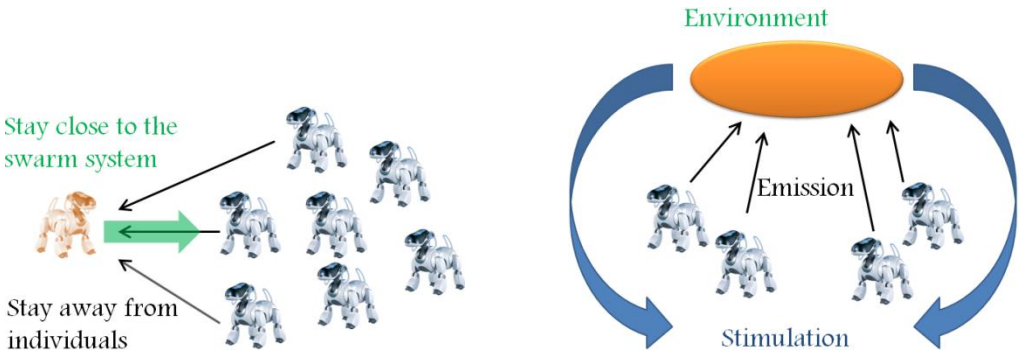


Figure 2.2. Two forms of swarm intelligence: (a) swarm-formation (b) stigmergy

Swarm robotics is a novel approach in coordination of mobile robot systems. It is inspired by observing existing interaction in animal societies (such as ants, wasps etc.). This method consists of a group of robots that interact and collaborate with each other to solve the problem. The system has interesting properties such as parallelism and redundancy. These features allow the system to be adaptive to changes in environment, robust to the failure of the robots and events, and shows good scalability when the problem size or number of robots increases. These three properties: robustness, flexibility and scalability have been stated as the motivation for swarm robotics:

1) **Robustness:** swarm system should be able to operate even after the failures in some individuals and disturbances in the environment. Several factors are considered for robustness of the system; First, redundancy in the system; that means malfunction or failure of individuals can be replaced by other entities in the swarm system. Second, coordination in the system should be decentralized; thus, destroying a single part of the system will not seize the overall operation of the system. Third, simplicity of the entities involved in swarm system; in

comparison with a single complex system that can perform the same task, individuals involved in swarm robotics would be simpler, this property make the system less prone to failures in individuals. Fourth, distribution of sensing; distributed sensing by individuals increase the amount of signal-to-noise ratio of the system.

2) **Flexibility:** if swarm robotic system is required to perform different tasks, flexible swarm robotic system should be able to assign the tasks to individuals. In other words, it must be able to create a modularized solution to different tasks.
3) **Scalability:** this requires that the swarm robotic system should operate even after increasing in the number of robots involved in the swarm system. The coordination mechanism between entities should be almost undistributed after changing in the number of robots.

Swarm robotic has fascinated a significant number of scientists and researchers and many research groups in top universities are working in this area. Currently, some of the research groups are operating in the following universities: MIT, Carnegie Mellon, Caltech, University of Alberta, Tokyo institute of technology etc.

2.4 Swarm optimization

As roboticists have progressed in making swarm of robots, biologists and other scientists interested in swarming behavior have made significant works in optimization. Optimization methods introduced in this area, run asynchronously and in a decentralized manner, so the word “swarm optimization” is appropriate to describe the concept. In this section, some basic concepts in optimization problems are introduced and after that some examples of such algorithms are provided.

2.4.1 Approach to the optimization problem

Classical search algorithms such as Depth-first search (DFS), Breadth-first search (BFS) and A* algorithms are designed to investigate search spaces in a systematical manner. The agent employed in search task achieves the systematic approach by keeping track of one or more path in memory and recording the order of explored alternatives along the path. When a goal is found, the path to the goal is called a solution. In many problems, however, finding the goal itself is desired and the path to the goal is not important. Examples of these two types of problems are illustrated in figure 2.3.

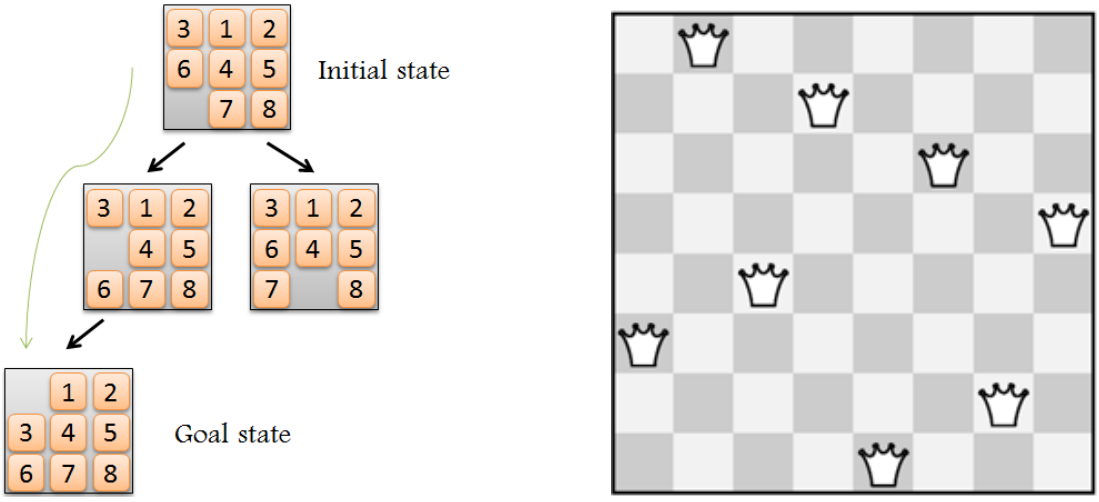


Figure 2.3. Two types of problems: (a) eight-puzzle problem is the problem of moving tiles on a square board until a certain arrangement of tiles (goal state) is achieved. The path to the goal is shown by green arrow. (b) eight-queen problem is the problem of placing eight queens on a chessboard in a way that no two queens attack each other. The path to this problem is irrelevant.

When the path to the goal is irrelevant, local search algorithms take into account. These algorithms are used in problems that can be formulated as minimizing or maximizing the value of an objective function (objective function is described later in this chapter) among a number of solution candidates. Algorithms in this category move from solution to solution in

the search space by searching locally, until an optimal solution is found or the considered time for search is over.

Local search algorithms have two important advantages: first, they only use a little memory. Second, they can find reasonable solutions in large or continuous search domains which systematic search methods are unsuitable.

To formalize an optimization problem, consider the following quadruple $< I, f, m, g >$ where:

- I represents a set of instances.
- $\forall x \in I, f(x)$ depicts a set of feasible solutions.
- $m(x, f(x))$ denotes the measure of $f(x)$ that is usually a positive real value.
- g is a goal function and is intended to be either minimization or maximization.

The goal is to find an *optimal solution* for x, in the following formula:

$$g = \arg_{x \in I} min/max \ m\big(x, f(x)\big) \quad (2.1$$

The above formula represents *objective function* in optimization problems and $m(x, f(x))$ is called *fitness function.* The fitness function measures the quality of the solution candidates and is problem dependent. In this study the fitness function is defined as following:

$$0 < fitness = \frac{\sum_{j=1}^{n} d_j}{\sum_{i=1}^{N} p_i} < 1 \quad (2.2$$

where:

- $P = \{p_1, p_2 \dots p_N\}$ is a set of pixels in the image captured by the camera.
- $D = \{d_1, d_2 \dots d_n\}$ is a set of pixels of goal object in the image captured by the camera.

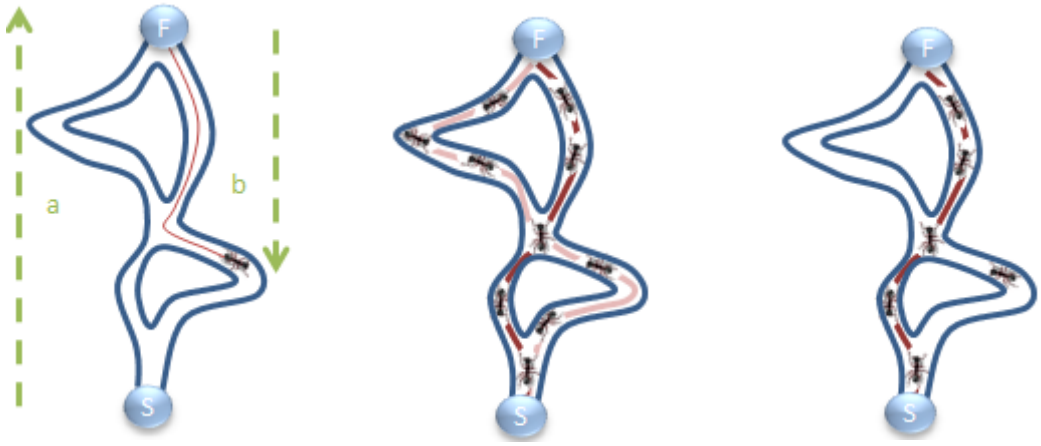
2.4.2 Examples of swarm optimization in the nature

In this section, three examples of natural optimization occurring in the nature are described. Scientists invented new optimization methods by mimicking the interaction mechanism in swarm of organisms.

2.4.2.1 Ant colony optimization

In the real world, ants wander around the environments, and after finding food, they return to their colony and leave pheromone trails on the path, which helps other ants to find such a path that ends up with food. Over time, the pheromone starts to evaporate, thus it loses its attractive strength. The more time required for an ant to travel down the path and back again, the more time pheromone trails have to evaporate. A shorter path, can be traveled faster, thus the pheromone density on this path remains higher in comparison with the longer path. The advantage of evaporation of pheromone is to avoid in converging into a locally optimal solution. Thus, if there were no evaporation at all, the path traveled by the first ant would remain attractive for being traveled by the others. When an ant finds a better path (i.e. shorter path), other ants are more likely to travel this path, and eventually others by leaving positive feedback on this path will lead all the ants follow this single path. An example of this type of swarming is shown in figure 2.4.

Ant colony optimization (ACO) method inspired by interaction of ants. The method is used in those challenges that involve finding paths to goals. Simulating agents achieve optimal solution by modifying parameters that represent all possible solutions. During experiments, agents record their position and its corresponded quality value. These values help the agents to locate better path in next iteration of the algorithm.



(a) (b) (c)

Figure 2.4. An example of ant colony optimization: (a) the first ant moves randomly around the nest (S) and after finding food at F it returns to the nest and leaves pheromone trails on the path. (b) Ants follow four possible ways, and pheromone trails on shorter path remain higher in comparison with the longer ones, thus it is more attractive for the ants to be traveled. (c) Over time, ants take the shorter path and other paths lose their pheromone trails.

2.4.2.2 Intelligent Water Drop

Shah Hosseini in [13] introduces a new method to solve optimization problem and called it "Intelligent Water Drops" or IWD algorithm. The method inspired by the natural mechanism in creation of natural river systems. The mechanism includes action and reactions between water drops in the river and occurrences of the changes in the environment that leads the river to flow. The idea is that drops of water in a river try to make an optimum path to achieve their goal (ending up with a lake or sea).

While water drops are moving in a river, they change the environment along the path that they flow. Moreover, the environment itself has a major role in creation of river's path. To understand the role of the environment in creation of the river path, consider the following scenario: drops of water in the river are flowing from a higher terrain toward a lower terrain and eventually join a lake. The path of the river is full of twists and turns along it (Figure 2.5 shows a river full of turns and twists along it.). During movement of the drops, gravity force pulls them toward the center of the earth. If there were no obstacles or barriers they would take the straight path, which is the shortest one, to reach the lake. However, obstacles and

barriers that are spread over the environment will lead the final path to be full of turns and twists. In contrast, drops of water always try to change their path to make a shorter path to achieve the lake. As time goes by, drops of water transfer the soils and obstacles along their path to another place to create short cuts. Also, while soils on the path are moving from a higher terrain to a lower terrain, this process would create a deeper area along the path that can hold a large volume of waters which may attracts more waters to flow.

Figure 2.5. Path of the Mississippi river that is created over a long time.

The method introduced in [13] proposes a problem solving technique inspired by this mechanism and an incomplete list of its application includes Travel salesman problem, Vehicle routing, N-Queen puzzle, Data clustering and Automatic multilevel thresholding.

2.4.2.3 Genetic algorithm

Genetic algorithm (GA) is another optimization method that is inspired by the process of natural evolution. In this method, a population of strings is selected so that these strings encode candidate solutions to the problem. Over time, these candidate solutions are manipulated to evolve and create a better solution to the problem.

A random population of candidate solution is created at the beginning of the search method. In each iteration, fitness of individuals involved in the population is evaluated. Then, some of the individuals are stochastically selected and modified to create a new population of candidate solution for the next iteration of the algorithm. Modification of the individuals is done through either recombining the individuals or randomly mutating them. This modification is shown in figure 2.6.

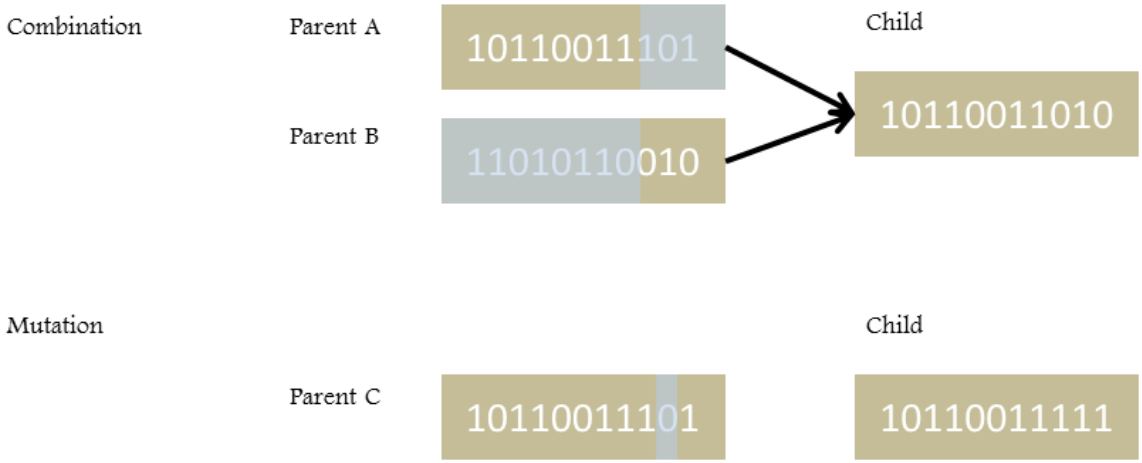


Figure 2.6. Two types of modifying candidate solutions in Genetic algorithm.

2.4.3 Particle swarm optimization

James Kennedy and Russell Eberhart in [7] introduce particle swarm methodology to optimize continuous nonlinear functions. The method has many similarities with evolutionary computation methods such as Genetic Algorithm (GA). The algorithm is classified as metaheuristic as it makes few or no assumptions in optimization problem and is able to search a large space of candidate solutions. However, the method is not complete, so it does not guarantee to find an optimal solution.

The behavior of particles in PSO is intended to simulate the behavior of bird flocking. To understand the idea of PSO, consider the scenario that follows: a group of birds are flying around an area and looking for food. Only one piece of food exists in the search area and all

the birds have no idea where the food is. But the birds are aware that how far the food is. So, an effective strategy for a bird is to follow the one that is closest to the food.

PSO learned from this interaction model in birds and is designed to deal with optimization problems. Each particle in PSO algorithm simulates bird behavior in search space. At any time, all the particles involved in swarm system evaluate the fitness value, which is intended to be optimized, and also they have velocities that are corresponded to the direction of movement. The particles move around the search space by following the current optimum particle.

A population of particles in particle swarm optimization (PSO) is initialized with random solutions and searches for optima by generating new solutions. In every iteration of PSO, every particle updates two values: First, the best solution and its coordination it has achieved so far. This is a local value and is called *pbest*. Second, the best solution and its coordination that has been observed by swarm system. This is a global value and is called *gbest*. After updating the gbest and pbest values, the particle calculates the next position to go and its velocity with the following formula:

$$v_{i+1} = \omega v_i + \phi_p r_p (p_i - x_i) + \phi_g r_g (g - x_i) \quad (2.3$$

$$x_{i+1} = x_i + v_{i+1} \quad (2.4$$

where:

- $x_i \in \mathbb{R}^n$ is the position of particle in the search-space at ith iteration.
- $v_i \in \mathbb{R}^n$ is the velocity of particle at ith iteration.
- p_i is the location of best solution has been achieved by the particle.
- g_i is the location of best solution has been achieved by swarm system.

- r_p and r_g are random numbers picked up from (0,1) range.
- ϕ_p, ϕ_g and ω must be studied for the system and are selected by experimenting the behavior performance of PSO method in swarm system. In this thesis we use the parameters studied by Bratton *et al.* in [5].

Figure 2.7 illustrates the calculations of x_{i+1} and v_{i+1}.

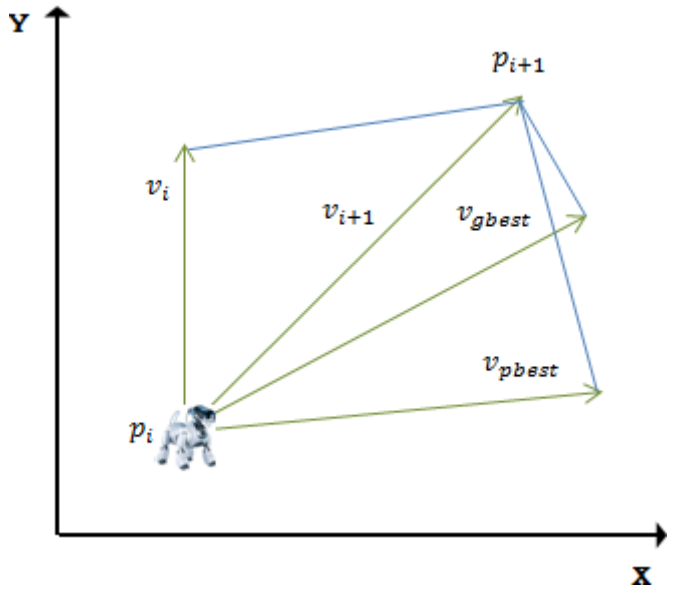
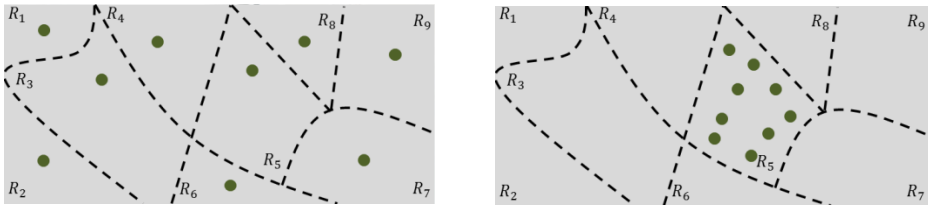


Figure 2.7. Calculation of particle's velocity and next position to go: at the end of i[th] iteration the particle is located at p_i position. v_{i+1} is calculated via summation over perturbation in v_i, v_{gbest} and v_{pbest}. p_{i+1} is calculated via summation of p_i and v_{i+1}.

2.4.4 Exploration vs. exploitation

Any method is able to solve a search problem in an infinite time. But, in real world experiments resources (including: time, memory etc.) are limited. Every intelligent search method must use a combination of exploration in new regions of the search space and exploitation in regions with potential solutions. To improve the performance of the search method, we need to create a suitable balance between exploration and exploitation. If the algorithm only explores in a search space, it leads in random search whereas if it only exploits in a search space, it leads in local search. These two concepts are illustrated in figure 2.8.



(a) (b)

Figure 2.8. The figures show the difference between exploration and exploitation. Search space is divided into 9 regions: $S = \{R_1 \dots R_9\}$ and green circles show the candidate solutions. (a) Candidate solutions are selected from different regions. Too much exploration leads to random search. (b) All candidate solutions are selected from a region. Too much exploitation leads to local search.

At the beginning of the PSO iteration, exploration is preferred. As the search progresses, it is required to search further in promising regions. One of the shortcomings of the PSO method is its requirement to the knowledge of the search space to create an efficient balance between exploration and exploitation. To reach an efficient balance between exploration and exploitation, many scientists invented new methods by hybridizing original PSO algorithm.

The first hybridization of PSO was introduced in [4] by Angeline (1998). In this paper a new population selection technique is used. The particles are distinguished as either "good" or "bad". Good particles are reproduced by mutation whereas bad particles are removed. Angeline claims that PSO method can benefit from this population selection technique.

Robinson *et al.* (2002) in [15] worked on optimizing profiled corrugate horn antenna. They hybridized PSO algorithm with Genetic Algorithm (GA). Their method switches from one method to the other after a number of iterations. They found that switching from PSO to GA (PSO-GA) outperforms GA, PSO and GA-PSO. They observed that PSO effectively explores in the search space while GA can effectively exploits in the promising regions.

Vesterstroom *et al.* (2002) in [14] introduce another hybrid method of PSO algorithm. They use the idea of division of labor which was studied earlier on insect swarm algorithms. In their method, if a particle does not improve after a number of iteration of algorithm, it exploits in the search space. To exploit in the search space, particle goes to the swarm system

best position with a new velocity vector which is selected randomly. Division of labor modifies the behavior of particle and intends to enhance the overall result. The improvement is achieved in unimodal problems whereas on multimodal function, the improvement was not significant.

Poli and Stephens (2004) in [12] introduced a hybrid method of PSO and a hill climber. The motion of the particles in the swarm system is confined to be on the fitness landscape. The particles slide on a fitness landscape, instead of flying over it. A particle in this method does not require a memory and it does not keep track of personal best. Exploration in the search space is guided by physics of masses and forces. Forces are: Gravity, springs and friction. Gravity enables the exploitation of the search space and spring provides exploration ability. Friction is used to slow down the search and focus on it.

In [11] Holden and Freitas (2005) hybridized PSO with ACO algorithm for classification purpose. Their method was used in functional classification of enzymes. They show that their new method have a better performance on a challenging biological data set in comparison with original PSO method.

2.5 Multi-robot path planning

In multi-robot systems, path planning and coordination mechanism are two important challenging issues that must be deal with them. The issues address the problem of how mobile robots share their understanding of workspace, while avoiding interference with each other's path, to enhance the overall system's performance. In this chapter, localization, which is a necessary tool for a robot to be able to plan a path, is discussed. Moreover, coordination mechanism of robots involved in swarm system is explained in next chapter.

2.5.1 Localization

Position estimation is one of the main problems in navigation of mobile robot systems in indoor environments. Probabilistic approaches are among promising candidates that provide real-time solution for robot's localization task. In this study, Monte Carlo Localization technique is utilized for localization purpose.

Two important problems in mobile robot systems are global position estimation and keeping track of local position. Global position estimation is the problem of determining the position of a robot on a previously learned map. Once global position estimation is done, local position tracking is the problem of keeping track of robot's position on the map over time. If the map of the environment is not available, several applications are developed that allow the robot to create a map as it explores the environment.

Dellaert *et al.* in [10] introduced Monte Carlo Localization (MCL) method and they used different approaches to represent uncertainties. In some of the classical methods, uncertainty is represented as a density function whereas in this method, uncertainty is represented by a set of samples that are generated randomly.

Using sampling-based approach has the following advantages:

1) The method can be used in global localization task. (In contrast to Kalman filtering approach that is unable to represent multi-modal distributions.)
2) It uses lesser memory in comparison with grid-based Markov localization technique.
3) Its accuracy is higher than the Markov localization method with a fixed cell size.
4) The method simplifies the implementation.

2.5.1.1 Robot localization

In robot localization task, the problem is to find the state of the robot at time-step *k*, where the knowledge from the initial state of the robot to the current state is given:

$$Z^k = \{z_k, i = 1 \dots k\} \quad (2.5$$

where:

- z_k represents knowledge of the robot at time-step k.
- Z^kindicates the state of the robot at time-step k.

In other words, the goal is to find the orientation of the robot in the following state vector:

$$X = [x, y, \theta]^T \quad (2.6$$

where:

- x and y show the position of the robot in the map.
- θindicates the heading of the robot.

The estimation problem is formulated as an example of Bayesian filtering problem, where the posterior density $p(x_k|Z^k)$ of current state of the robot depends on all measurements. To localize a robot, the posterior density $p(x_k|Z^k)$ needs to be calculated at each time-step. This computation is done in a prediction phase and an update phase.

Prediction phase: in this phase, a motion model is created to predict the current state of the robot in the form of predictive probability density function $p(x_k|Z^{k-1})$. We assume that the current state depends only on the previous state (and not on the sequence of previous events) and a set of control commands u_{k-1}. Motion model is represented as following conditional density:

$$p(x_k|x_{k-1}, u_{k-1}) \quad (2.7$$

Finally, predictive probability density function can be calculated from following formula:

$$p(x_k|Z^{k-1}) = \int p(x_k|x_{k-1}, u_{k-1}) p(x_{k-1}|Z^{k-1}) dx_{k-1} \quad (2.8$$

Update phase: in this phase a measurement model is created to utilize the information received from sensors to calculate the posterior density function $p(x_k|Z^k)$. We assume that z_k is independent of previous measurements Z^{k-1} given x_k. Moreover, measurement model is given in form of likelihood $p(z_k|x_k)$. This shows the likelihood of the robot to be at location x_k, when z_k is observed. The posterior density function is formulated through Bayes theorem in the following manner:

$$p(x_k|Z^k) = \frac{p(z_k|x_k) p(x_k|Z^{k-1})}{p(z_k|Z^{k-1})} \quad (2.9$$

This process is repeated recursively and the knowledge about the initial state is assumed available in a density form of $p(x_0)$. In global localization, $p(x_0)$ might be a uniform density over all possible positions of the robot in the map. Moreover, in tracking localization, initial position is evaluated as the mean and covariance of a Gaussian around x_0 .

2.5.1.2 Monte Carlo Localization

In sampling-based methods, density $p(x_k|Z^k)$ is represented by a set of random samples in the following manner:

$$S_k = \{s_k^i; i = 1 \dots N\} \quad (2.10$$

Where:

- N is the number of samples.
- s_k^i is the i[th] sample generated at time-step k.

By using histogram or kernel based density estimation methods, we can approximately create the density from the samples.

The goal is to calculate the set of samples S_k that is drawn from $p(x_k|Z^k)$. To accomplish this task, MCL-method utilizes following phases:

Prediction phase: in this phase, the motion model is applied to each sample ${s^i}_{k-1}$ by sampling from $p(x_k|{s^i}_{k-1}, u_{k-1})$. Thus, a set of ${S'}_k$ is drawn that approximates the random samples from density $p(x_k|{s^i}_{k-1}, u_{k-1})$. ${S'}_k$ does not incorporate any information received from robot's sensor.

Update phase: in this phase, observation of z_k is taken into account. Moreover, each sample in ${S'}_k$ is assigned with a weight ${m^i}_k = p(z_k|s_k'^i)$, in other words, likelihood of $s_k'^i$ when z_k is given. S_k is obtained by drawing samples from ${S'}_k$ weighted set.

This algorithm is performed in $O(N)$ time, where N is the number of samples. Thus, allows the robot to perform real-time localization. An example of one iteration of this algorithm is shown in figure 2.9. In the figure, panels in the top row represent the exact density and the panels in the below row represent the particle-based representation of the density.

In column A, a cloud of particles in S_{k-1} shows the uncertainty about robot's position. In column B, the robot is told to be moved 1 meter from the previous time-step. In column C, sensors observe a landmark, half a meter away in top-right corner. Eventually in column D, the effect of resampling process from the weighted set is shown.

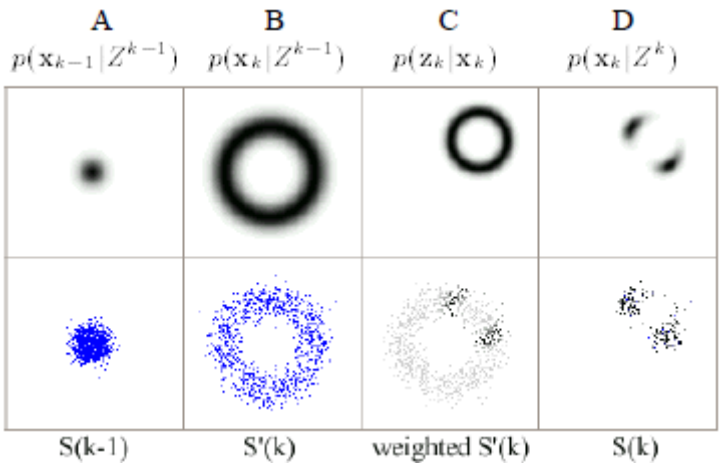


Figure 2.9. Probability densities and particle sets for one iteration of MCL-method [10].

2.5.2 Sensors

Sensors are critical part of the autonomous robots. They allow a robot to contact with outside environment. There are different types of sensors that are used in different tasks such as detecting obstacles, collision avoidance, etc. In this section, the sensors that are used in localization tasks are briefly described:

2.5.2.1 Sonar system

All the robots involved in swarm system of this study are equipped with sonar systems. Sonar systems are divided into arrays, and each array contains eight sonars. Sonar array's transducers are multiplexed and at any time, only one disc is activated per array. The sonar ranging acquisition rate is fixed to 25 Hertz (40 milliseconds per transducer). And sonars are able to sense obstacles ranging from 10 cm up to 5 meters. The positions of the sonar systems in the PatrolBot are shown in figure 2.10.

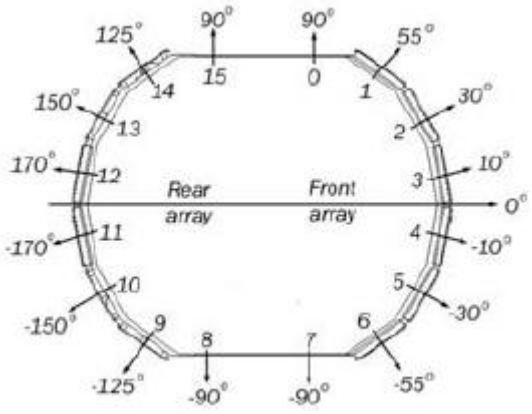


Figure 2.10. PatrolBot's sonar arrays. Sonars are numbered clockwise from 0 to 16.

2.5.2.2 Laser range Finder

PatrolBot is equipped with SICK LMS200 laser range finder (LRF) system and its precision is almost 25 times better than sonar system. LRF is able to perform 180 readings in 180 degrees, whereas sonar system has one reading for every eight degrees. Laser beam is sufficiently focused which leads to the least absorption and modification by reflecting medium. The laser senses objects from 5 cm to 50 m with a ranging accuracy of ± 18 mm. High accuracy of the LRF system allows the robot to perform fine localization and object avoidance tasks. A picture of the LRF system is shown in figure 2.11.

2.5.2.3 Gyroscope

A gyroscope is a mechanical tool for keeping track of changes in orientation of a robot. Wheel-rotation sensors and gyroscope are used in a dead-reckoning method for estimating the heading and position of the robot on the map. PeopleBot is equipped with an analog gyroscope system that helps the robot to correct rotational errors (such as unpredictable wheel slipping, sliding, etc.).

Figure 2.11. SICK LMS200 laser range finder (LRF) system

Having reviewed of the basics, in next section we move onto details of the technical approach to the solution.

3 Technical approach to the solution

To study the performance and efficiency of the algorithms discussed in chapter two, a mobile robot system is designed to simulate the algorithms behavior in real world environment. In this chapter, details of the technical approach are described.

3.1 Map of the environment

Map of the robots' environment is stored as a text document. This document consists of lines and points that represent both obstacles and map boundaries in the environment. The document is mostly used to assist the robots in navigation and localization tasks. Maps of the environment are created by PatrolBot robot (specifications of the robots are explained in the experimental results section) which is equipped with range-finding sensor inside the robot.

To create the map, PatrolBot either wanders around the environment or moves under human supervision. While robot is moving, it uses laser system to detect and collect data about sensible environment such as obstacles, walls and etc. These data are stored either as points or as lines in the document along with their relationship to each other. Other information of the environment, which is also stored in the map, is as follows: number of lines and points in the map, forbidden areas, map resolution, and etc. Figure 3.1 shows an example of the map of the environment.

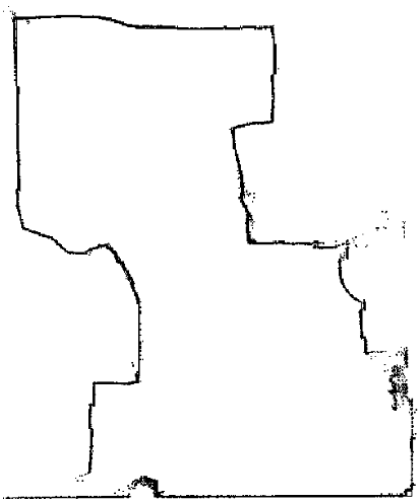


Figure 3.1. Map of the room 225, Bourns College of engineering EBU 2, at University of California, Riverside. Boundaries of the search environment are shown as points or lines.

3.2 Discretization of the map

In real world, robots might encounter some problems in movement and path planning tasks. For example, consider the following scenario: to search an area, the robot plans a path to go to a position in the search environment. The position is very close to the wall (e.g. the distance of the point to the wall is about 10 mm), although some parameters in robot's path planning system are set to avoid the collision, yet if the robot plans to go to this position it will increase the probability of a collision. One efficient strategy is to move the robot to a safer point that is close enough to the desired point to search for the target. To help the robot to move in *safe* regions, map of the search space is discretized.

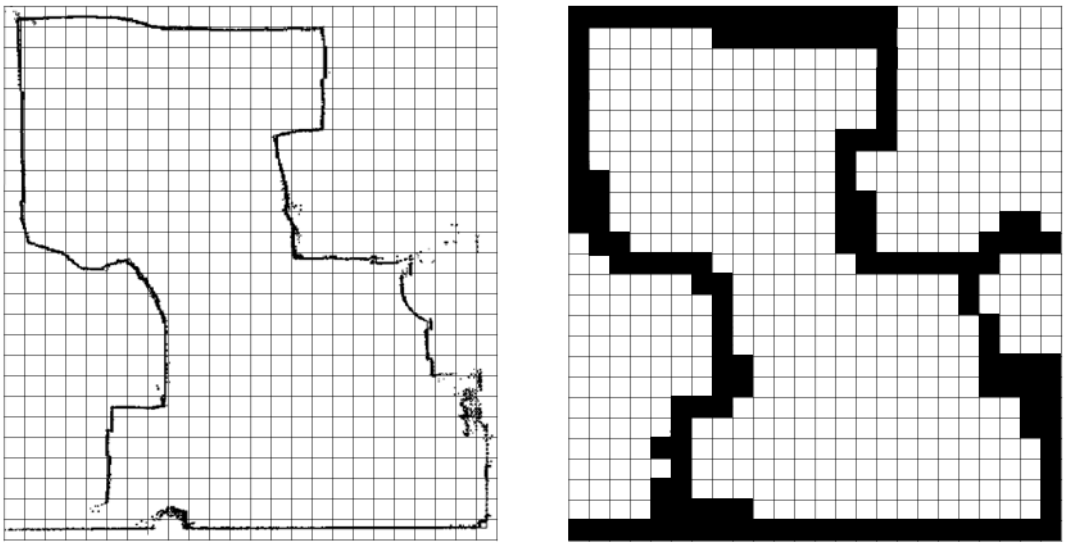
Discretization is the process of converting continuous search space into discrete counterparts. In discretization, map is divided into non-overlapping square areas that are also called units. If a search space and its area are respectively denoted by S and A_s, after discretization of the map, the units can represent the search space in the following manner:

$$S = \left\{ U_i \middle| A_s = \sum_{i=0}^{N} A_{u_i} ; and\ \forall U_i, U_j\ where\ i \neq j\ U_i \cap U_j = \emptyset \right\} \quad (3.1$$

where:

- U_i represents the i[th] unit obtained after discretization of the map.
- Moreover, A_{u_i} denotes the area of the unit U_i.

After discretization, any unit that overlaps with points or lines on the map will be marked as *unsafe unit* and the rest of the units are considered as *safe units*. Figure 3.2 shows a map after discretization process.



(a) (b)

Figure 3.2. Discretizing the search space: (a) the map is discretized into non-overlapping units which are represented by squares (b) those units that overlap the points or lines are marked as unsafe cells and are shown by filled units.

In path planning and robot navigation, the robot plans the path and moves in the search space only through the safe cells and not the unsafe cells. The unit's size must be small enough so that when it is marked as unsafe unit because of a single point inside it, this must not lead to marking a large area of the search space as unsafe area. A suitable size of the unit is studied in experimental result section.

3.3 Selecting points of interests on the map

While the robot is moving around the search space, regions of the map that are close enough to the robot and can be observed in the field of view of the robot's camera are marked as *visited regions.* To organize the search regions on the map, points of interest on the map are selected; the centers of the safe units are considered as points of interest. If the robot marks the center of the unit as visited, then the entire unit is considered as visited unit. This strategy reduces the number of revisited regions on the map and prevents the robot from looking for the target in a region that is almost impossible to find a target. Points of interest are shown in figure 3.3.

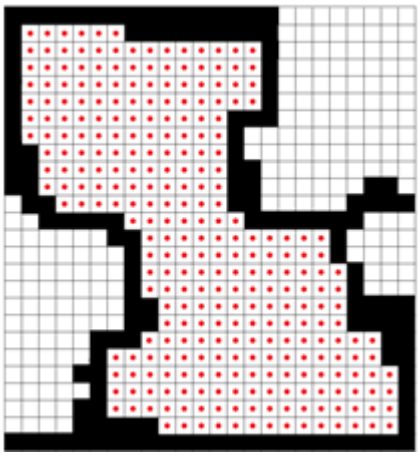
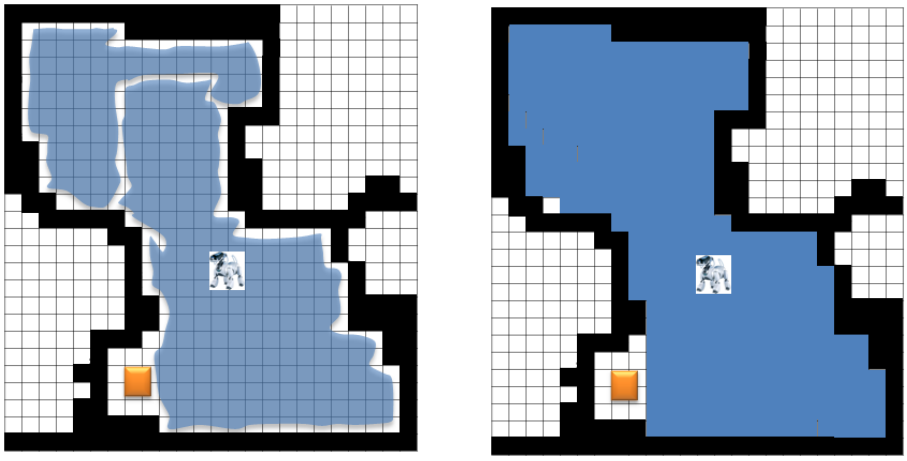


Figure 3.3. Points of interest (red dots) are located at the center of the safe units.

To understand the application of this strategy, consider the following scenario: at some point in the search experiment, most of the search space is visited. There are a lot of small areas on the map that haven't been visited and only one large unvisited area exists which contains the target. The robot is about to go to a new position in the map and look for the target. Since it is almost impossible to find a target in small areas, which are smaller than the target, an efficient strategy is to mark those small areas as visited to remove them from the search list. Figure 3.4 illustrates this scenario.



(a) (b)

Figure 3.4. Marking units as visited. Visited regions are marked by blue color and orange box is the target. (a) A lot of small areas on the map are left as not visited regions. (b) Those units that their centers have been visited are marked as visited areas. Thus, only a few not visited areas containing the target are left.

Another factor that must be considered in selecting a suitable size for the units is the size of the target that we are looking for. Size of the unit must be small enough so that if its center is visited, the entire unit can be marked as visited area without losing a high chance of finding a target.

3.4 Object recognition

To search for the target in the environment and evaluate the objective function value, an object recognition system is used. While the robot is moving in the search space, object recognition system processes the images captured by camera and if it finds the target in the image, it evaluates the objective function value; otherwise it will return zero. To simplify the recognition process, the target used in search experiments has a unique color in search environment.

3.4.1 Removing irrelevant pixels

At the first stage of the object recognition system, those pixels that have different color from the target are removed from an image. To detect desired color (color of the target) on the image, a similar method to the one introduced by Gao *et al.* in [9] is used.

Redness, greenness and blueness of each pixel are determined with the following formulas:

$$r(x,y) = \frac{2R_i}{G_i + B_i} \; if \; (G_i + B_i > 0) \tag{3.2}$$

$$g(x,y) = \frac{2G_i}{R_i + B_i} \; if \; (R_i + B_i > 0) \tag{3.3}$$

$$b(x,y) = \frac{2B_i}{G_i + R_i} \; if \; (G_i + R_i > 0) \tag{3.4}$$

where:

- $r(x,y), g(x,y)$ and $b(x,y)$ represent redness, greenness and blueness of the pixel at (x,y) position in the image.
- (R_i, G_i, B_i) is color vector of ith pixel that is projected in RGB color space.

Table 3.1 shows redness, greenness and blueness of some colors.

Color	**Redness**	**Blueness**	**Greenness**
Pink	1.29	0.83	0.90
Indian red	6.5	0.22	0.31
Violet	1.29	0.54	1.29
Light blue	0.84	0.98	1.19
Cold gray	0.93	1.04	1.01
Aqua	0	2	2

Table 3.1. Redness, Greenness and Blueness values for some colors.

After calculating these values for each pixel in the image, those pixels that their redness, blueness and greenness values are out of a proper range from the values for the target, will be removed from the image. Figure 3.5 shows an example of this process.

3.4.2 Connected component labeling

After removing irrelevant pixels on the image, some blobs of pixels might remain on the image. We use connected-component labeling to distinguish different objects on the image.

Connected component labeling or blob extraction is one of the applications of graph theory, where subsets of connected regions are labeled to distinguish between different objects. This method is used in computer vision to detect connected components in a binary digital image. A graph (containing sets of vertices and edges) is constructed from an input image; the edges indicate connected *neighbors* and vertices store information, which is used in comparison heuristic. The algorithm traverses the graph and labels the vertices based on the connectivity and corresponded values of their neighbors.

Two-pass algorithm is one of the algorithms in this area. It is simple to implement and works on 2-dimensional binary data. It has two passes over the image: in the first pass, it assigns temporary labels to the pixels on the image, and in the second pass, it updates the value of labels by its equivalent class.

While traversing over the pixels in the image, connectivity check is carried out for every pixel. 4-connectivity condition only checks North and West (e.g. up and left) neighbors of current pixel. The conditions that are checked for each pixel (in 4-connectivity condition) are as following:

1. Does the left pixel (West) have the same label?
    - If yes, then current pixel is in the same region, thus assign the same label to it.
    - If no, check the next condition.
2. Do the up and left pixels have the same value but they have different labels?
    - If yes, then the up and left pixels are in the same region, so the regions must be merged and the pixels in those regions (including the current one) must be labeled with a unique number.
    - If no, check the next condition.
3. Does the left pixel have a different value, but the up pixel have the same value?
    - If yes, assign the label of up pixel to the current one.
    - If no, check the next condition.
4. Do the up and left pixels have different values?
    - If yes, assign a new label id to the current pixel.

Through this method, different labels assigned to the pixels of each object can distinguish different objects in the image. Figure 3.5 illustrates different stages of object recognition mechanism.

3.5 Coordination mechanism

While robots are moving around the search space to find the target, they exchange two important pieces of data about the search experiment with each other: visited units and gBest. When a robot visits a point of interest, it lets the other robots know about it by sending a packet containing the location of the POI over the TCP/IP network. Upon receiving the information about the visited unit, robot marks the corresponded unit as visited and eliminates the unit from search list. This process prevents a unit to be selected for search more than once.

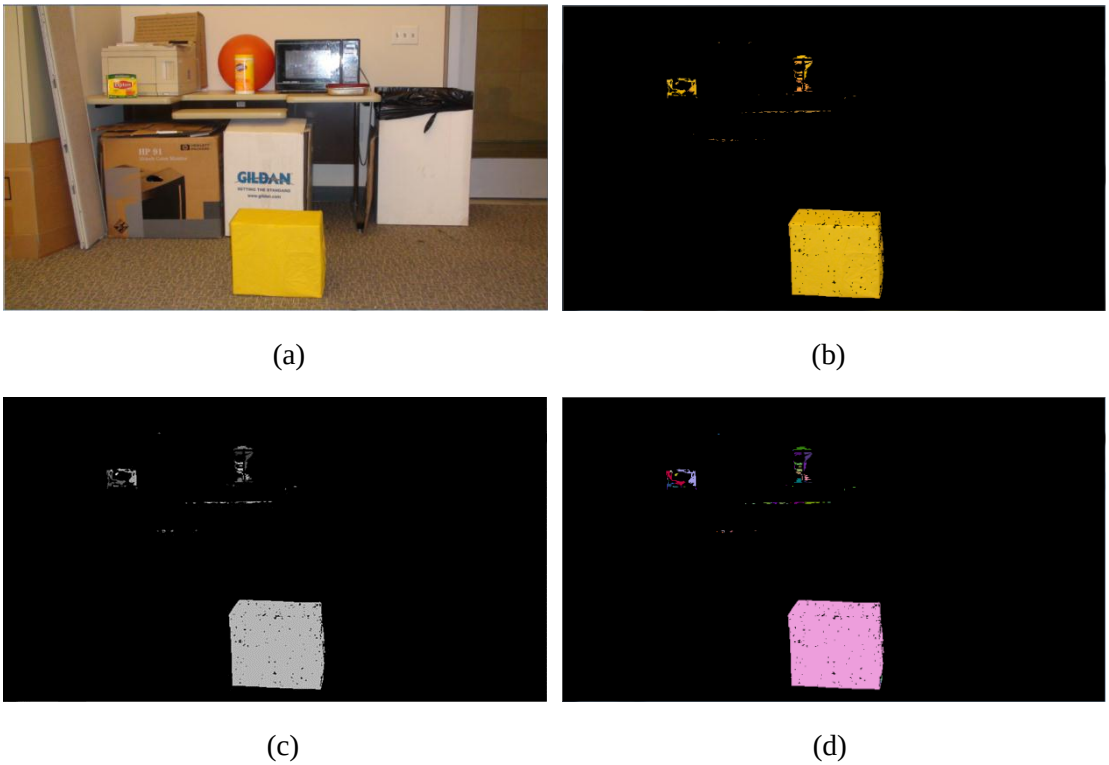


Figure 3.5. Different stages of object recognition system: (a) the original image is captured by the camera, the goal is to find the target (yellow box) in the image, though some objects in the image have the same color as target. (b) Those pixels that have irrelevant redness, blueness and greenness values are removed from the image. (c)The image is converted into an equivalent black-and-white image. (d) After labeling connected-components, different colors are assigned to different component in the image and the target is colored in pink.

CurrentFitness, gBest and pBest are vectors that represent fitness and its corresponded position respectively for current, global and local fitness. When a robot finds a better fitness value, it updates its pBest and gBest values to the current fitness and lets the other robot know about it. Figure 3.6 shows the pseudo code of updating fitness values.

```
• If currentFitness > pBest
        pBest = currentFitness
• If pBest > gBest
        gBest = pBest
        send out the new gBest to
        the other robots
```

Figure 3.6. Pseudo code of updating fitness values.

An example of this coordination is shown in figure 3.7. The figure is a snapshot of the simulator running on PeopleBot during an experiment. The green points and lines in the picture represent the map of the environment. Red points are points of interests that are not visited yet, whereas white points are points of interests that already have been visited by either PeopleBot or P3AT robot. P3AT's location on the map is shown with a green circle and PeopleBot's location is indicated by a blue circle. A white triangle ahead of PeopleBot shows the field of view of the robot's camera. When a red point falls into the field of view of a camera, it will be marked as visited.

In experiment shown in figure 3.7, both P3AT and PeopleBot have exchanged the information about visited cells to each other. Thus, PeopleBot or P3AT will never select any cell that is visited by another robot in the future. And as you can see from the figure, from the start time to since the time snapshot is taken, PeopleBot's current and best local fitness is always zero, but its best global fitness value is 0.0035 and is received from P3AT.

3.6 Search strategy

During experiments, robots select a new position in the search space to look for the target. The following methods are used as position selector algorithms: Exhaustive search, Genetic algorithm, Particle swarm optimization and MPSO. When a candidate position is selected by a search method, a nearest unvisited POI is selected for the robot to be searched. In this section, search strategies are reviewed.

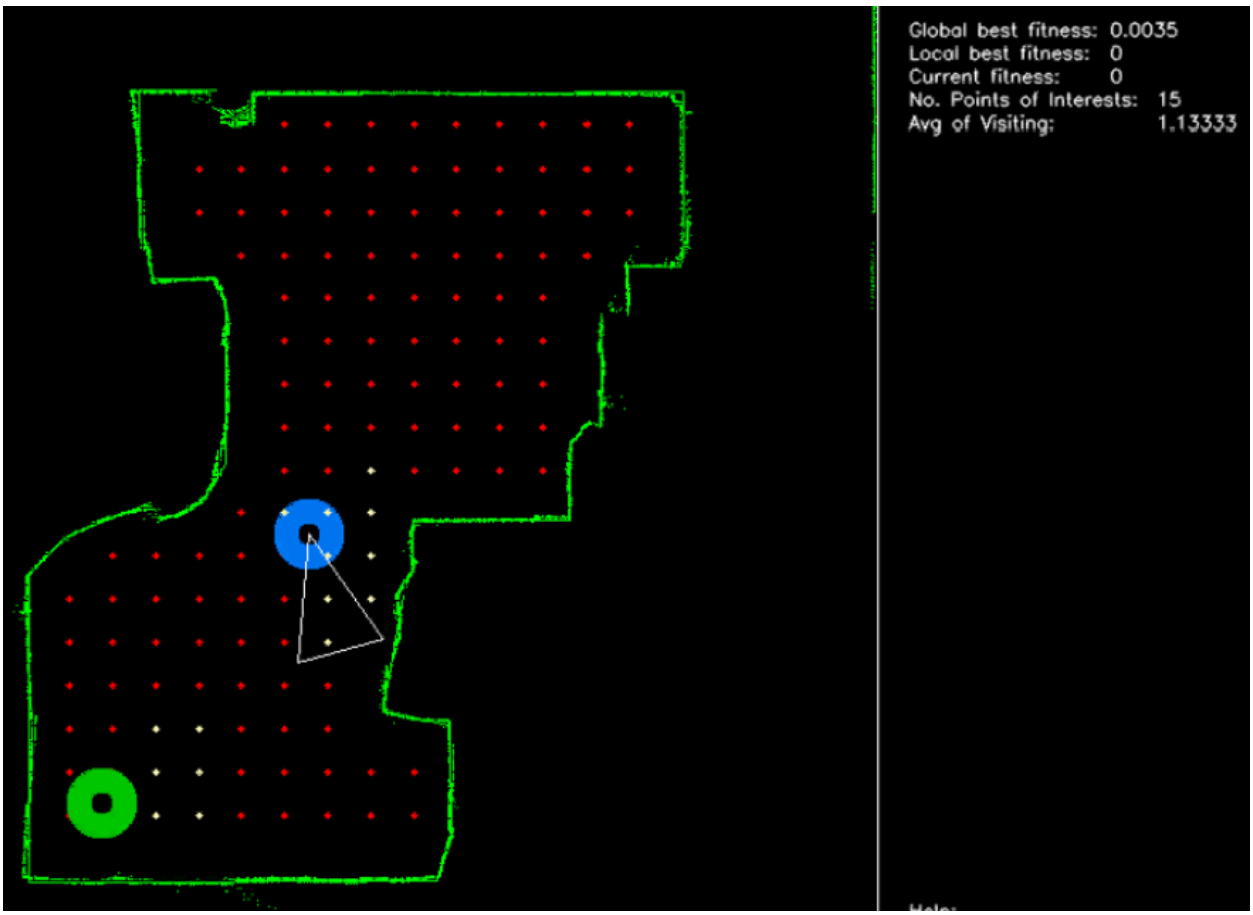


Figure 3.7. Simulator running on PeopleBot robot. Map boundaries are shown in green points and lines. PeopleBot and P3At are shown in blue and green circles respectively. Some of the PeopleBot's variables are shown on upper right section of the figure. Best global fitness variable for PeopleBot is updated by P3AT and both robots are exchanging their visited cells.

3.6.1 Exhaustive search

Exhaustive search is a base-line search method to measure the performance of the other search algorithms. In this algorithm, robot moves among unvisited units one by one until either it finds the target or one of the robots in the swarm system finds the target. If more than one robot is utilized in an experiment to find the target, they will divide the search space into two sections with equal number of POIs. Then, each robot will be assigned to one of the sections and at any time if one of the robots is done with searching in its own section, it will help the other robots to search for the target in their assigned sections.

During the search time, robots will exchange the information of visited cells, but they do not exchange the best global fitness. Since exhaustive search contains no learning tool in finding

the target, therefore the robots does not care whether the fitness is 0.99 or 0.01 and will never change the order of the units that is going to visit them.

The order of visiting units is determined by their priorities. The priorities of the units are shown in figure 3.7. Upper units have the highest priorities to be visited and depending on the direction that can be set to left or right, units on left or right have next priorities. Finally, if all the up, left and right units are visited the robot will decide to search in down units. Figure 3.8 shows an example of exhaustive search.

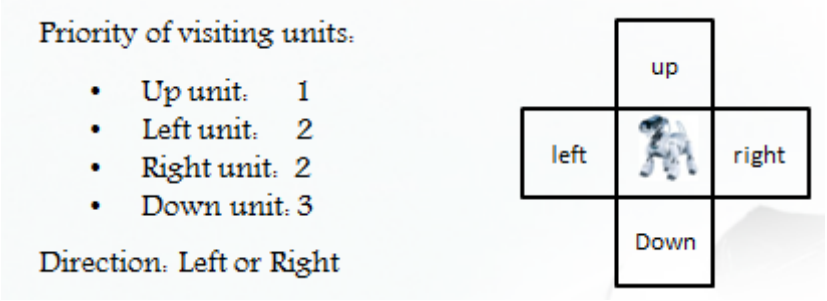


Figure 3.7. Priority of units in exhaustive search.

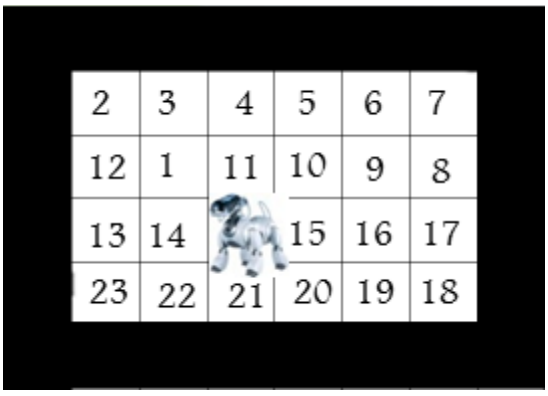


Figure 3.8. An example of exhaustive search: the numbers in the units indicate the order that the unit is going to be visited.

3.6.2 PSO search

As explained in section 2.4.3, PSO method calculates the next position to be searched by robot through the following formula:

$$v_{i+1} = \omega v_i + \phi_p r_p (p_i - x_i) + \phi_g r_g (g - x_i) \quad (3.5$$

$$x_{i+1} = x_i + v_{i+1} \quad (3.6$$

In this formula, ω, ϕ_g and ϕ_p are coefficients that must be selected through experimenting the behavior efficiency of PSO method in swarm system, whereas, $v_i, (p_i - x_i)$ and $(g - x_i)$ vectors are studied during the experiments.

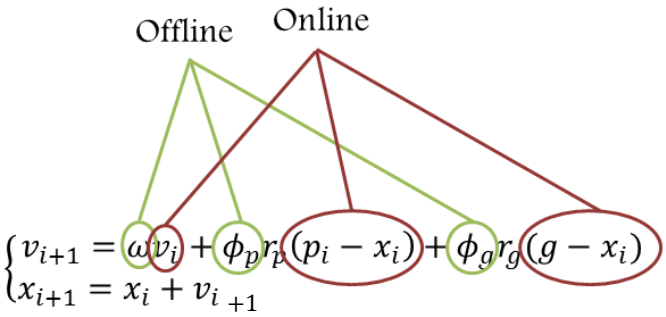


Figure 3.9. ω, ϕ_g and ϕ_p coefficients are studied before experiments. $v_i, (p_i - x_i)$ and $(g - x_i)$ vectors are learned during the experiments.

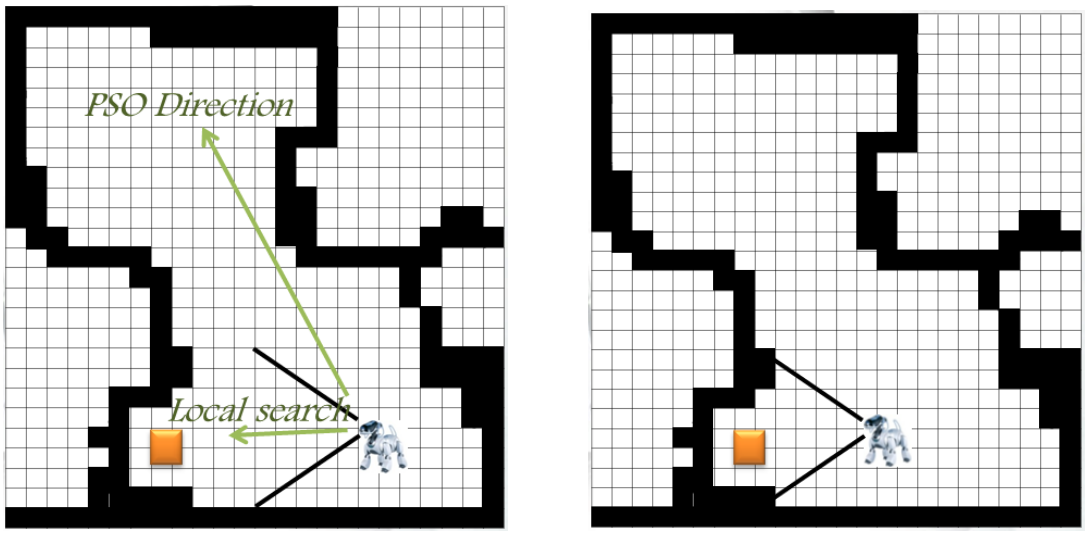
Coordination mechanism of the robots involved in an experiment is described in section 3.5 and pseudo- code of the algorithm is shown in figure 3.10.

3.6.3 MPSO search

In this section, PSO method that is described in 3.6.1 is modified to improve its performance. Utilizing original PSO method in swarm system has two disadvantages. First, in some cases although the current fitness value is high enough to be a strong evidence for existence of target in robot's neighborhood, original PSO guides the robot to search for the target in a position that leads the robot to increase its distance to the target. To prevent this situation, PSO is modified so that if current fitness value goes beyond half of the goal fitness the robot will start local search instead of working with PSO method. An example of this situation is shown in figure 3.11.

For each Agent i = 1 … N
Do
Initialize the best local position: $p_i \leftarrow x_i$
Initialize the best global position: $g \sim U(b_{lo},\ b_{up})$ where: $b_{lo} and\ b_{up}$ indicate the boundaries of search space
Initialize the particle's velocity: $v_i \sim U(-|b_{lo} - b_{up}|, |b_{lo} - b_{up}|)$
Calculate the POIs of the search space: $S = \{u_1 \dots u_n\}$, n is the number of POIs on the map
Initialize the set of visited units: $V = \emptyset$
End
While (Target is not found && some units are not visited)
Do
For Each Agent I = 1 … N
Do
Pick up random numbers: $r_p\ and\ r_g \sim U(0,1)$
Update the particle's velocity and next position to go: $v'_{i+1}\ and\ x'_i$
Select the nearest unvisited POI to the new position to go.
While(agent has not achieved the selected POI)
Do
Take a photo by camera
Evaluate the current fitness value: c_i
If $(c_i > p_i)$ Then
Update p_i
if $(p_i > g)$ Then
Update g and send out the new best global fitness to other agents
Mark the units that fall into camera's field of view as visited
Update visited set: V
End
End

Figure 3.10. Pseudo-code of the PSO algorithm

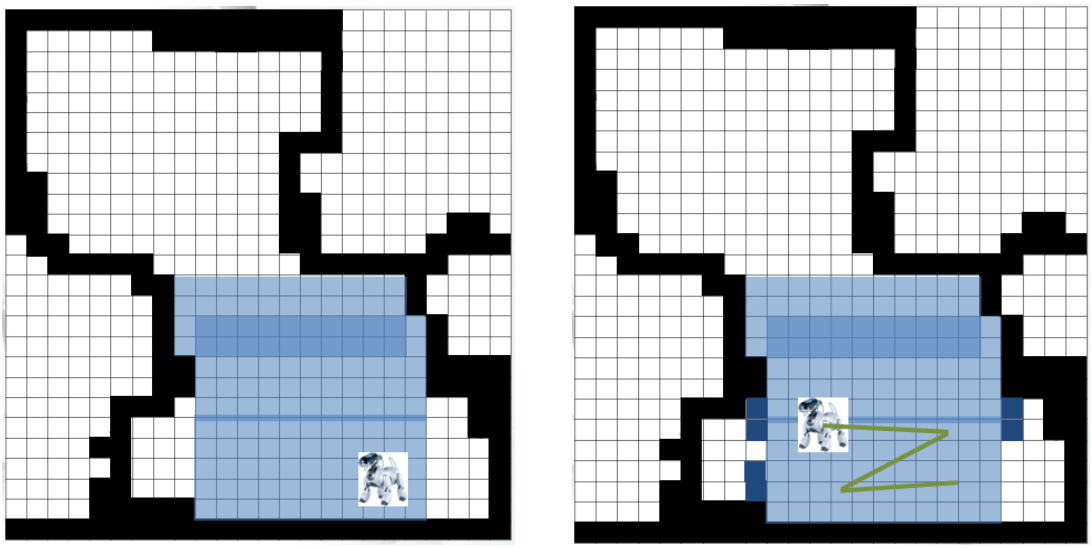


(a) **(b)**

Figure 3.11. An example of first disadvantage of the original PSO method: (a) at current position of the robot the fitness value is 62% of the goal fitness but for next position to go, PSO evaluates an irrelevant position. (b) Instead of using PSO method to evaluate next position to go, by performing local search robot is able to find the target.

Second, in some cases, when the best global fitness value is not significant to be considered as symptom of the target, the PSO method guides the robot to move back and forth mostly

between the visited units rather than visiting unvisited units. Revisiting visited units will increase the search time and decreases the efficiency of the algorithm. Figure 3.12 shows an example of this situation.



(a) (b)

Figure 3.12. An example of second disadvantage of the PSO method: (a) those units that are colored in light blue are visited units. (b) After next three iteration of the PSO method, the robot mostly moves among visited units (green lines show the path that was traveled by the robot), rather than visiting unvisited units (units that are colored in dark blue).

To prevent this situation from happening, every time that PSO method selects a point to visit, we will check to see if the number of the visited units on the path is less than the half of the length of the path, then the robot will work with PSO method. Otherwise, if the number of the visited units on the path is equal to or greater than the half of the length of the path, then the robot will perform local search. This technique prevents a robot to revisit visited units and can enhance the performance of the algorithm. Figure 3.13 shows an example of this decision-making situation.

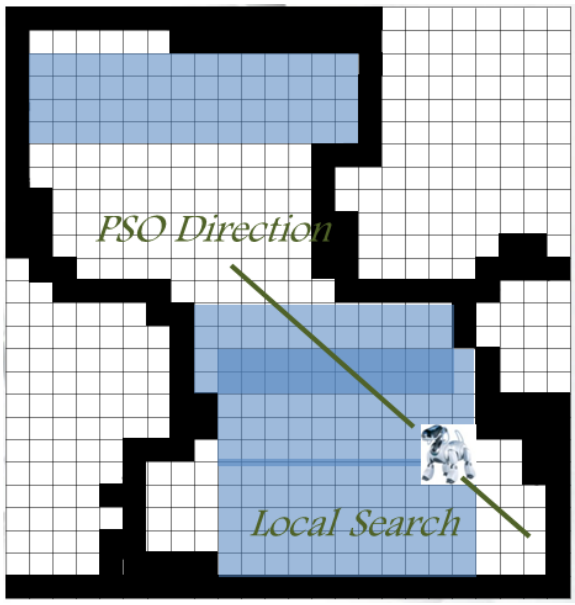


Figure 3.13. Length of the path calculated by PSO method contains 14 units in total; 12 of them are visited and only 2 of them are unvisited. In MPSO, Instead of working with PSO, robot decides to visit two unvisited local units rather than traveling on the path that results in revisiting visited units.

3.6.4 Genetic algorithm

The idea of utilizing genetic algorithm (GA) is to employ evolution in the computer. GA utilizes the basis of selection and evolution to create candidate solution to the problems. The evolution occurs through the crossover and mutation mechanism. We have used coordination mechanism between the robots in swarm system similar as what we have used for PSO and MPSO methods. GA is designed to work well in any environment and figure 3.14 shows the pseudo-code of GA.

Termination condition in the main loop can occur in the following conditions:

- An individual finds a desired fitness.
- Enough number of iterations has passed.

Computation of the averaging in the pseudo-code shows the overall progress of the swarm system and this computation is not absolutely necessary.

Now that the search algorithms and coordination mechanisms are described we will move on to the experiments section of this study.

```
// Initialization phase
Select initial population
Evaluate fitness of each individual
Compute the average fitness of the population
//Main loop
While (Terminating condition)
        Select individual that has best fitness value to reproduce
        Mate pairs of individual at random
        Apply crossover
        Apply mutation
        Evaluate fitness of each individual
        Compute the average fitness of the population
End
```

Figure 3.14. Pseudo-code of the genetic algorithm

4 Experimental results

In this chapter, performance of the algorithms and behavior of the swarm system are studied in different conditions; using combinations of three different robots, two different targets and two different maps are used in conducting the experiments.

4.1 Experiment Environment

In this section, specifications of elements which are used in experiments are described:

4.1.1 Robot Specifications

Combinations of P3-AT, PeopleBot and PatrolBot robot are deployed in swarm system to study the behavior and performance of the algorithms. Details of their specifications follow:

4.1.1.1 Pioneer 3-AT (P3-AT)

P3-AT is a four wheel drive robotic system and is manufactured by MobileRobot Company. It is a popular robot to perform outdoor activity. The robot has an embedded computer which enables the robot to perform image processing itself. It carries 3 swappable batteries and communication is Ethernet-based. It contains 8 forward and 8 rear sonar sensors that can detect the obstacles at any range from 15 cm to 7 m. The robot can reach the speed of 8 meters per second. Mapping, navigation, monitoring, vision and cooperation are important applications of P3-AT. The robot is shown in figure 4.1.

Some of the other important specifications of the robot follow:

- The robot's integrated PC is a common EBX form-factor board.
- It is equipped with a Cannon VC-C50i camera and maximum frame rate of the camera is 30 frames per second.

- The robot is able to plan paths and autonomously navigate through the environment.
- It mostly uses sonar data for localization task.

Figure 4.1. P3-AT robot

4.1.1.2 PeopleBot

PeopleBot is also made by MobileRobot Company. It can be used for both object and target tracking. PoepleBot features similar architecture as P3-AT platform and has a chest-level extension which facilitates its interaction with people. It is shown in figure 4.2 and some of its important features follow:

- It is equipped with infrared sensors that enable the robot to detect tables when approaching it.
- The robot is equipped with Bumblebee®2 CCD camera and the camera's frame rate is 20 frames per second.
- It is equipped with gyroscopic sensors that increase rotational accuracy. Gyro corrections and odometry-based position estimator incorporate in localization task.

- The robot uses high-speed, high-torque and reversible-DC motors. Each of the motors equipped with a high-resolution optical quadrature shaft encoder to estimate speed, change in position and advanced dead-reckoning.
- The robot is equipped with bumpers to sense the collision with obstacles when other sensing tools (such as sonar and infrared systems) fail to detect the objects.

Figure 4.2. PeopleBot robot

4.1.1.3 PatrolBot

PatrolBot is an autonomous service robot which is also built by MobileRobots Inc. The robot is able to be used in several positions such as search robot, delivery robot and security robot. The robot is shown in figure 4.3 and some of the important specifications of the robot follow:

- The robot is equipped with SICK LMS200 laser range-finding sensor which enables the robot to scan buildings, navigate autonomously and create floor plans.

- The robot is equipped with a Cannon VC-C50i video camera and its frame rate is 30 frames per second.
- In addition to the laser, the robot is also equipped with 8 forward and 8 rear sonar sensors. These sensor systems are used for navigation and localization purposes.
- The robot is equipped with a Wi-Fi system that enables the robot to be controlled remotely and operate autonomously.

Figure 4.3. PatrolBot robot

4.1.2 Target

In experiments, the goal is to find the target. The target is hidden randomly in the search space and the robots move around the search space to find the target. Two targets that are used interchangeably are shown in figure 4.4.

(a) **(b)**

Figure 4.4. Targets of the experiments: (a) red pillow target (b) yellow box target

The running time of the experiments starts when the robots start moving around the search space to find the target and it ends when at least one of the robots can see 3000 pixels of the target through the camera. Robot can see 3000 pixels of a target when the target is located approximately 2 meters ahead of the robot.

4.1.3 Environment

Two different search spaces are considered to conduct the experiments. Most of the experiments are done at room 225 of Engineering Building unit 2 (EBU II) at University of California, Riverside, and some of the experiments that measures the scalability and robustness of the system are conducted at courtyard of EBU II of the same university. Figure 4.5 shows two search areas.

Figure 4.5. Map of the search areas that is located on second floor of the EBU II at University of California, Riverside. The area that is colored in light red is the map of the room 225 and the area that is colored in blue is the map of the courtyard

Map of the search spaces are created by PeopleBot robot. Room 225 has a size of 11860 mm × 12020 mm which is about 142.557 m^2, and size of the courtyard is about 946.744 m^2. Some parts of the both of the search spaces are occupied with obstacles such as tables, trash cans etc.

4.2 Experiments

Several experiments carried out to test the performance of the search algorithms (PSO, MPSO and Exhaustive) against each other and to test the properties of swarm system. To avoid repeating the description of the test cases the term '*Scenario N*' is used to refer to the N^{th} test case after its description.

4.2.1 Unit size

As explained earlier in technical approach section, size of the units must be small enough so that when it is marked as unsafe unit because of a single point (an object overlapping the unit) inside it, this must not lead to marking a large area of the search space as unsafe area. Because, by removing a large area of the search space, which contains the target, the chance of finding the target decreases. Moreover, size of the unit must be small enough so that if its center is visited, the entire unit can be marked as visited area without losing a high chance of finding a target in the rest of the unit area.

In contrast, if size of the units is large, then fewer numbers of points of interests will be chosen for search space, which leads to reduction of search time. Furthermore, by selecting big units for the search space, during experiments a number of unvisited small areas will be removed from search list that will help the robots to search for the target in promising regions.

Scenario 1:

In this scenario, PatrolBot robot is utilized to search exhaustively for the red pillow in one of the challenging initial positions for both robot and the target. Initial positions for PatrolBot and the target are shown in figure 4.6.

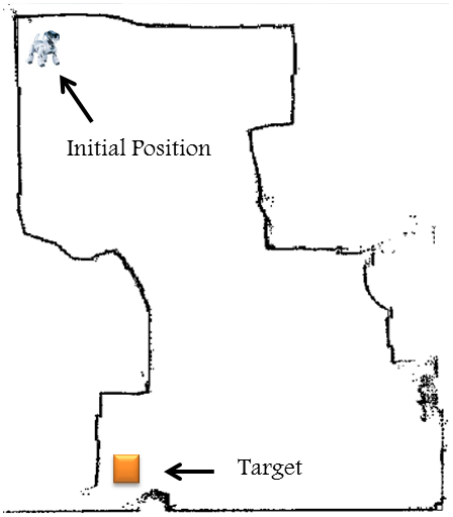


Figure 4.6. Initial positions of the target and the robot in scenario 1.

Different experiments are conducted to learn a suitable unit size. For each set of experiments, a unit size is selected from table 4.1 and experiments are repeated for 20 times. During the search experiment, if the robot finds the target, the experiment is considered successful, otherwise it is considered failure. Average of the search time and percentage of failure and success in finding the target is provided in table 4.2.

Unit size (cm)	Points of Interests
25	611
50	134
75	49
100	26

Table 4.1. Unit sizes and their corresponded number of POIs

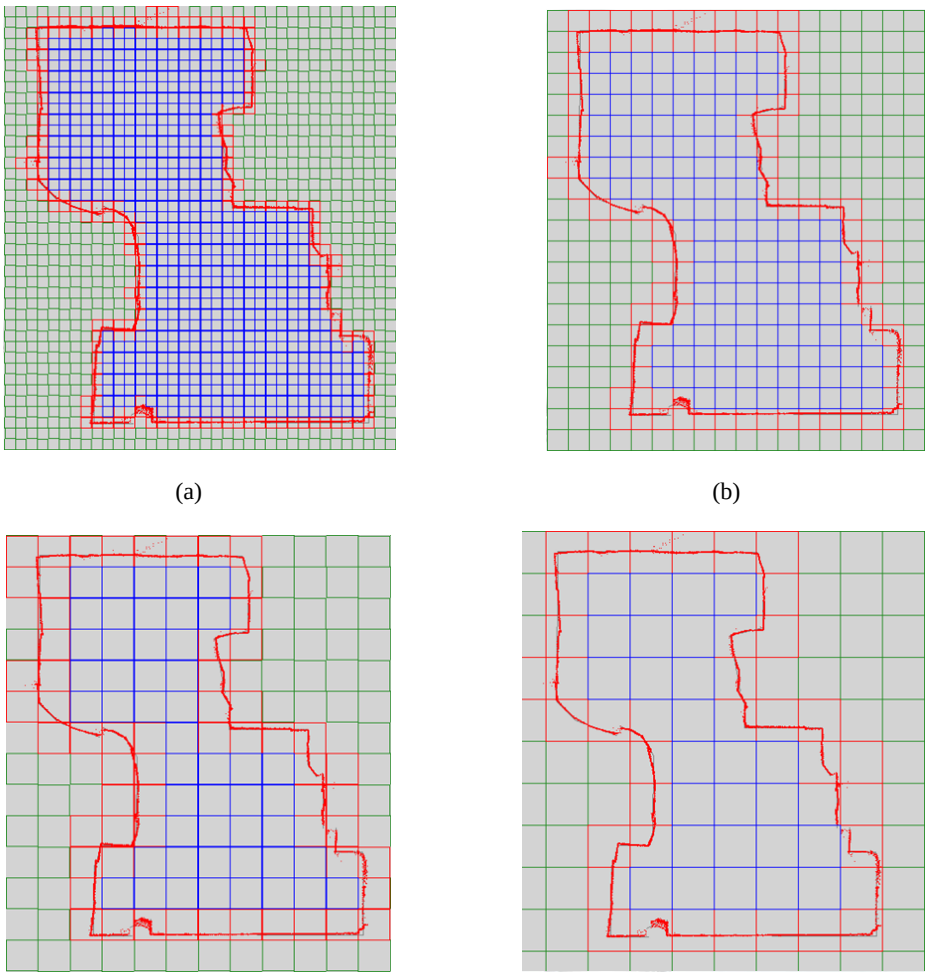


Figure 4.7. Discretization of the map with different unit sizes. Safe units are shown with blue squares, whereas red units indicate the unsafe cells. Points of interests are located at the center of the safe units. (a) 25 cm is selected as unit size (b) 50 cm is chosen for the size of the unit (c) 75 cm is selected as unit size (d) unit size is one meter.

Unit Size (cm)	POIs	Avg Time (Min)	Fail	Success
25	611	12:35	0	100%
50	134	9:36	0	100%
75	49	6:58	20%	80%
100	26	5:19	60%	40%

Table 4.2. Average time and number of failure and success in experiments.

Table 4.2 shows that as the unit size increases, the number of points of interests on the search space increases, therefore it increases the average time of the experiments and decreases the

accuracy of the robot to find the target. Based on the results in table 4.2, 50 cm is chosen as a suitable unit size because of its higher accuracy (in comparison with 75 cm and 100 cm unit sizes) and lesser average time to search the search space (in comparison with 25 cm unit size).

4.2.2 Performance of the algorithms

In this section, we study performance of algorithms in different conditions. First, performance of the algorithms is studied with respect to increasing the size of the search space. Second, we will see changes in search time by increasing the number of the robots in search experiments. Third, initial positions of the robots and the target are modified to see how they affect the search time. Finally, a larger search space is selected that shows a significant difference in perfromance of the algorithms.

4.2.2.1 Increasing size of the search space

In this section, performance of the algorithms (Exhaustive as a baseline, PSO and MPSO) are compared with each other when the size of the search space increases. Search time, average of revisiting visited units and number of visited units are used as the measurements to compare the performance of the algorithms. The idea is: when an algorithm searches in lesser search regions or revisits lesser visited regions while looking for a target in an experiment it will have a better performance.

Scenario 2:

In this scenario, search space (S) is divided into four different regions ($R_1, R_2, R_3 \; and \; R_4$) that is shown in figure 4.8. For each set of experiments a combination of search regions, are selected as search areas. Table 4.3 shows the size of the search areas and their corresponded number of POIs.

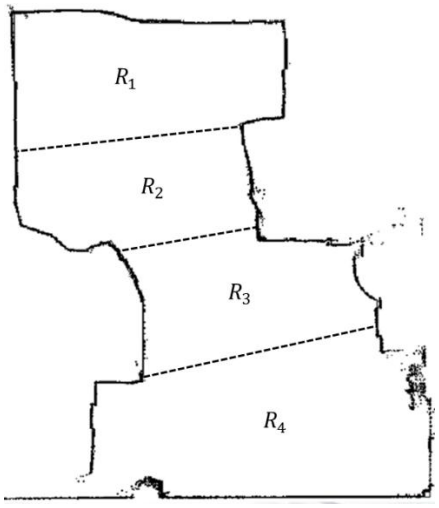


Figure 4.8. Dividing the search space into four different regions. During the experiments size of the search space grows from $\frac{1}{3}S$ to S.

Size of the search space	Number of POIs	Regions
$\frac{1}{3}S$	49	R_1
$\frac{1}{2}S$	73	$R_1 + R_2$
$\frac{2}{3}S$	97	$R_1 + R_2 + R_3$
S	145	$R_1 + R_2 + R_3 + R_4$

Table 4.3. Size of the search regions in experiments.

Single robot:

In this section, performance of the algorithms is compared with each other while using a single robot in each experiment. For each size of the search space, every robot is utilized to conduct the experiments 5 times. The initial positions for both target and the robot are shown in figure 4.9 and figure 4.10 compares the results of experiments.

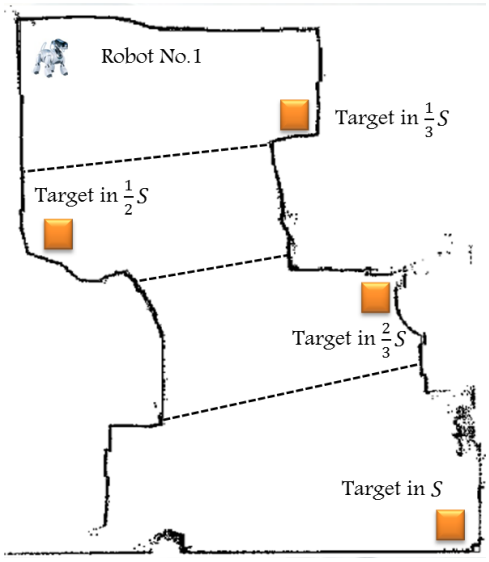


Figure 4.9. Initial positions of robot and target. Initial position for target changes based on the size of the map.

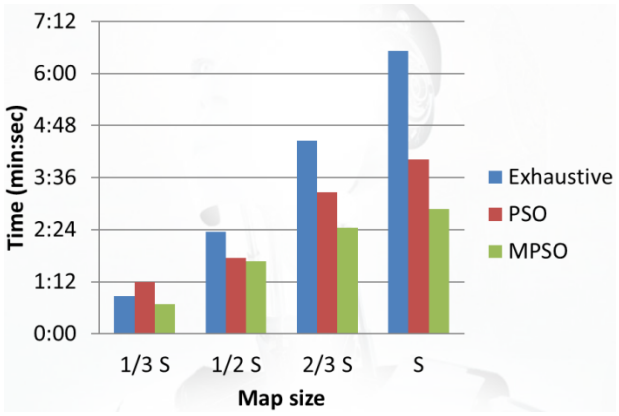
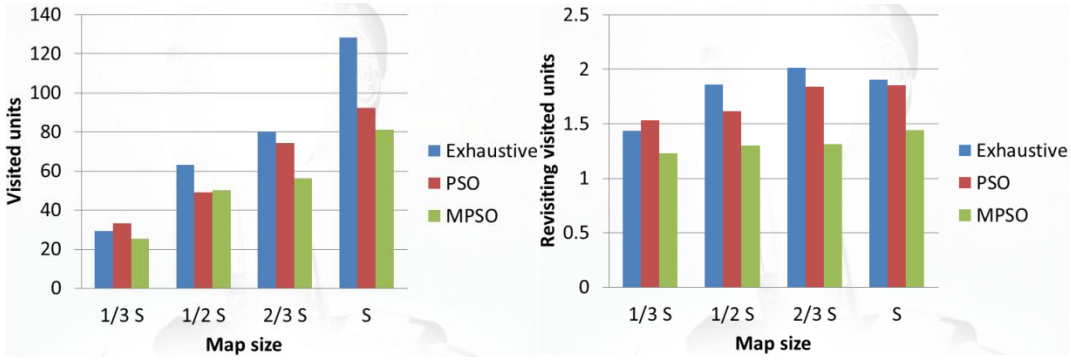


Figure 4.10. Performance of the algorithms in single robot experiment while increasing the size of the search space.

As we can see from figure 4.10, the running time of the algorithms increases with respect to the growth of the map size, and MPSO outperforms PSO and Exhaustive search. PSO method has a better performance in comparison with Exhaustive search except in $\frac{1}{3}S$ experiment.

Figure 4.11 shows two other measurements that affect the performance of the algorithms. Most of the time MPSO outperforms PSO and PSO outperforms the exhaustive search except in $\frac{1}{3}S$ for PSO and $\frac{1}{2}S$ for MPSO.



(a) (b)

Figure 4.11. Two measurements affecting the performance of the algorithms: (a) If a method guide the robot to find the target by visiting lesser search regions, it would have better performance. (b) If a method leads the robot to revisit visited regions while looking for the target, it would have worst performance.

Multi-robot:

In this section, multi-robot is used to see how it affects the performance of the algorithms. The initial positions for targets are similar to single robot test and for the second and third robot we consider new initial positions. Figure 4.12 shows the initial positions for multi-robot test.

For each number of robots (either two or three) utilized in search task, combinations of robots are used to conduct the experiment 5 times.

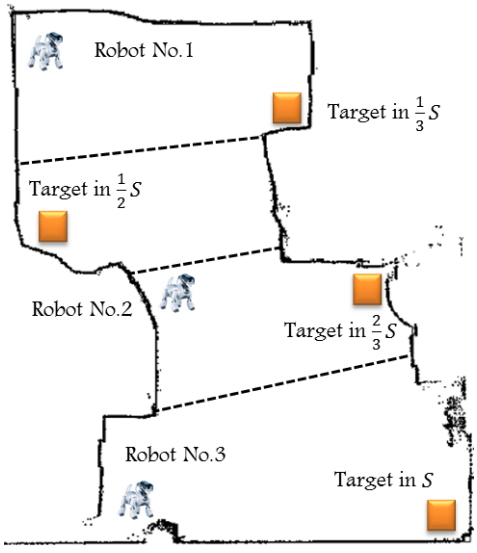
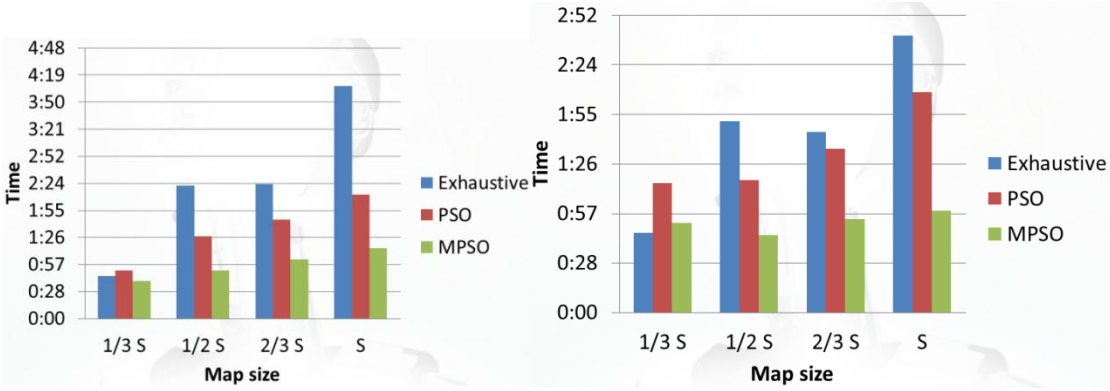


Figure 4.12. Initial positions of robots and target in multi-robot experiment

Utilizing two or three robots in search task has advantage of visiting more search regions in lesser time, however, robots involved in search tasks might interfere each other's path, which will increase the search time. Figure 4.13 compares the performance of the algorithms in multi-robot system.



(a) (b)

Figure 4.13. performance of the algorithms in multi-robot search: (a) Two robots are used in search task. (b) Three robots are used in search task.

Similar to single robot search, the running time of the search algorithms increases with respect to the growth of the map size. As we can see in figure 4.13, in general, MPSO outperforms PSO and PSO outperforms Exhaustive search except in $\frac{1}{3}S$ experiment. Advantage of utilizing multiple robots in search task is that in lesser time robots can search more of the search regions, whereas, disadvantage of using multiple robots is that the robot might interfere each other's path, which can increase the search time. In Exhaustive search, multiple robots divide the search regions into non-overlapping regions and each robot search in its region. This mechanism reduces the number of path interference of robots. Thus, exhaustive search have a better performance in $\frac{1}{3}S$ experiment.

4.2.2.2 Modifying initial positions

In this section, performance of the algorithms is studied under the condition of changing the initial positions of target and robots. Exhaustive method searches the search space in a systematical manner. Thus, in some combination of initial positions for target and the robots, Exhaustive search is able to find the target in a lesser time. We will see that although in some combinations of initial positions, performance of the methods changes, but in average MPSO performs better than Exhaustive and PSO. Moreover, in this section we will see that by using more robots in search task, overall performance of the system improves.

Scenario 3:

In this scenario, 10 different combinations of initial positions for robots and target are randomly selected for single robot search. In multi-robot search the same initial positions are used for target and first robot, and other random positions are chosen for the second and third robot. Figure 4.14 shows initial positions for this scenario.

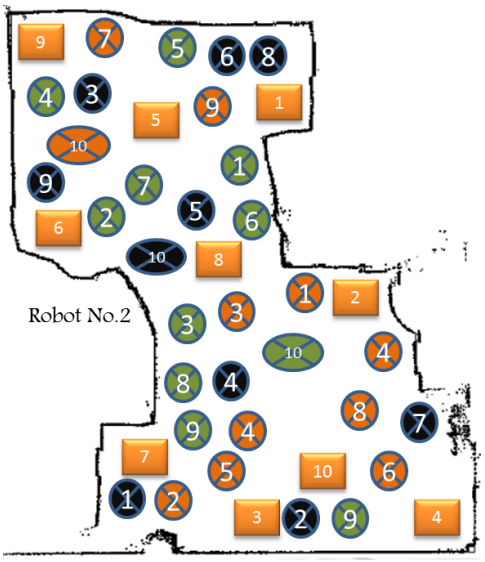


Figure 4.14. Initial positions of the robots and the target in scenario 3. Orange box represents target. First, second and third robot are depicted respectively by green, dark blue and orange circles.

Figure 4.15 shows the results of the experiments.

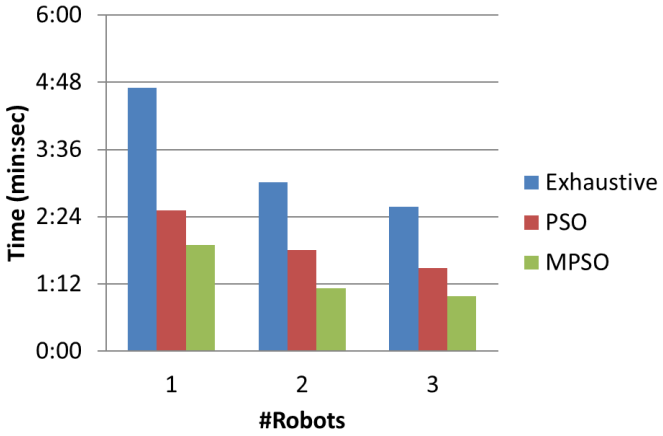


Figure 4.15. Result of the experiments described in scenario 2.

Figure 4.15 shows that MPSO outperforms both Exhaustive and PSO methods. Moreover, increasing the number of robots that are deployed in search task, improves performance of the system.

4.2.2.3 Changing the search space

In scenario 3, differences between performance of the system, when different number of robots are used in the search task, is just a couple of seconds which does not show significant improvement in deploying more robots for search task. In this section, we conduct more experiments in a new search space that is shown in figure 4.5. New search space is about 7 times larger than the previous search space.

Scenario 4:

In this scenario, a number of initial positions are selected for three robots and target. Then, experiments are conducted 10 times for utilizing one, two and three robots in search task.

Figure 4.16 shows the initial positions of the robots and the target for this scenario and figure 4.17 compares the performance of the algorithms in this scenario.

Figure 4.16. Initial positions of the robots and the target in scenario 4. Orange box represents the initial position for the target. Initial positions of the first, second and third robots are shown by dark blue, green and orange circles respectively.

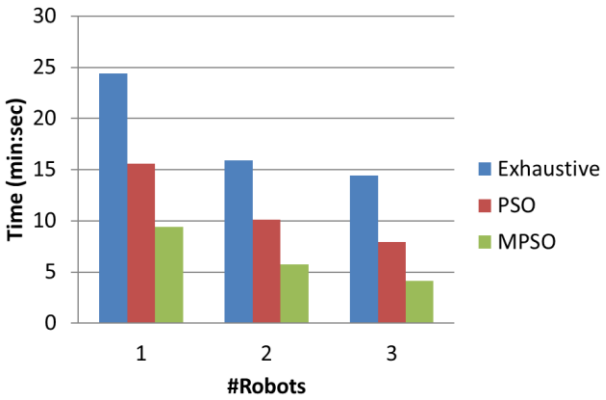


Figure 4.17. Performance of the methods in scenario 4.

Figure 4.17 shows that performance of the methods in new search space has similar results to scenario 2 and 4. In addition, utilizing more robots in search task significantly improves the performance.

4.2.2.4 Genetic algorithm

In this section, performance of the methods is compared with Genetic algorithm (that is described in section 3.6.2). For this purpose, similar condition in scenario 3 is set up for Genetic algorithm. Experiments are conducted 10 times for Genetic algorithm. Figure 4.18 shows the results of the experiments.

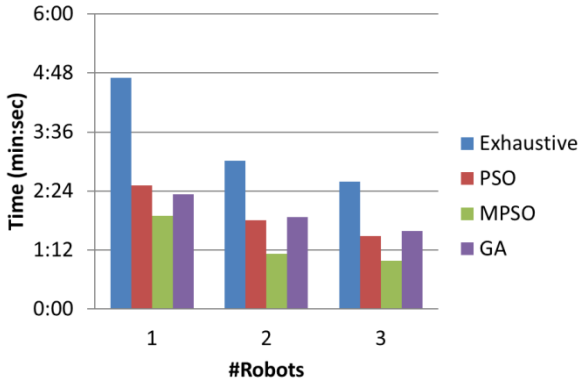


Figure 4.18. Performance of the Exhaustive, PSO and MPSO methods in comparison with GA.

As we can see from figure 4.18, MPSO always outperforms other methods and Exhaustive search as expected performs worse than the other methods. In single robot search, GA performs slightly better than the PSO, but in multi-robot search, PSO has a better performance.

4.2.3 Properties of the system

As we have discussed in chapter one, researchers design swarm intelligence systems to maintain following important properties: scalability, flexibility and robustness. We have discussed the scalability of the system in section 4.2.2.3 and since in search task we have declared only one task, thus we do not need to measure the flexibility of the system. In this section, we have conducted several experiments to measure the robustness of the system.

4.2.3.1 Moving objects

One of the important issues in performance of the system is the localization task. Although the robots are equipped with different sensors to perform localization task, in real world experiments, sometimes robots fail to move because of the localization's failure. In this section, the experiments in scenario 4 are repeated 10 times with the following modification: three people are wandering around the search space, while robots are looking for the target. This modification sometimes leads to the localization error and prevents the robots to find the target. Table 4.4 shows the result of the experiment.

#Experiments	#Objects	#Fail	#Success
10	3	20%	80%

Table 4.4. Results of the experiments for moving objects.

Table 4.4 shows that although the system fails to find the target in some experiments, in an acceptable number of experiments is able to accomplish the search task.

4.2.3.2 Robot failures

In this section, we have conducted several experiments to see when one or two robots fail during an experiment, the entire system is able to find the target or not. To study the robot failures, two sets of experiments are conducted. First, after 10 seconds of the beginning of the experiment we randomly stop one of the robots. Second, after 10 seconds of the beginning of the experiment two robots stop working. The initial start position of the robots and the target are similar to the scenario 4. Table 4.5 shows the results of the experiments.

#Experiments	#Robot failures	#Fail	#Success
10	1	10%	90%
10	2	30%	70%

Table 4.5. Results of the experiments for robot failures.

Table 4.5 shows that in most of the cases the swarm system is able to accomplish search task, even after the failures at some of the robots in the system.

4.2.3.3 Modifying environment

In this section, we modify the map of the search space by putting six random objects around the search space while the information about these objects is not stored in the map file. This modification leads to the error in robot's localization task and sometimes prevents the robots to find the target. The initial start positions for the robots and the target are similar to the scenario 4. Table 4.6 shows the result of this experiment.

#Experiments	#Random objects	#Fail	#Success
20	6	65%	35%

Table 4.6. Results of the experiments for modifying environment.

As we can see in table 4.6, the robots are sensitive to the modification of the map and modifications leads to the failure at localization task of the robots, which in most of the cases prevents the system to accomplish the search task.

5 Conclusions and future works

5.1 Conclusion

In experimental results section, we have conducted several tests to compare Exhaustive search, PSO, MPSO and Genetic algorithm with each other. Moreover, we have conducted more experiments to measure some important properties of the system such as robustness and scalability. In this section, based on our observation in our experiments we conclude followings:

1) Although in a few number of experiments performance of the methods changes, but in general PSO is 61% faster than Exhaustive search and MPSO is 57% faster than PSO method. Moreover, performance of the Genetic algorithm and PSO method is almost similar to each other. Thus, MPSO has a better performance between all the methods that we have studied in this thesis.
2) As we add more robots to the swarm system, the performance of the system increases. In average, if swarm system contains two robots to search for the target, the system 55% outperforms the single robot search and if we use three robots in swarm system, the system accomplishes the task 20% faster than the two robots.
3) Utilizing PSO method in multi-robot system has following advantages:
    a. Simplifies the implementation.
    b. Decentralizes the computation between individuals in the system.
    c. The system is robust to the changes in the environment.
    d. The swarm system is easily scalable.
4) Using the techniques of selecting POIs and discretizing the map of the environment have following advantages:

a. Through discretization we can convert a continuous search space into discrete counterparts.
b. Discretization organizes the search regions into a number of non-overlapping squares called unit.
c. They assist the robots to reduce the number of times that robots revisit visited regions.
d. They help the robots to look for the target in promising regions and eliminate some regions from search list where finding target is almost impossible.

5) Experimental results show that although in some combinations of the initial positions for both target and robots, performance of the methods changes, but in average MPSO works faster than the PSO which outperforms Exhaustive search.
6) During the experiments, robots need to plan a path to move in the search space. To accomplish this task they need to perform localization task to find their selves on the map. This localization task can be affected by modifying elements of the map and also moving objects which in some cases lead to the failure at localization task and prevents the system to find the target. Thus, equipping robots with more accurate localization sensors can help the system to maintain robustness property.

5.2 Future work

Since in most of the real world applications of utilizing robots to search for a target, the target is moving in the search space, one of the interesting future works for this thesis is to enable the system to track moving objects.

More studies can be performed in mechanism of selecting POIs to facilitate the search task to add more POIs in promising areas and eliminate some unnecessary POIs.

To facilitate the search task and visiting more unvisited units in a lesser amount of time, path planning system must be modified to consider visiting unvisited POIs as much as possible, while planning a path to a certain position.

Another important issue in our swarm system is to handle localization task with respect to the modifications in the search environment. In future works, if the robots are equipped with more accurate sensors, the system would be more robust.

Moreover, if we equip the robots with more advanced sensors and tools, they can be utilized in more search tasks. For example if equip the robots with special drills and sensors to detect water and oxygen, they can be used in moon exploration project to move on the surface of the moon and search for water and oxygen.

6 Bibliography

[1] S., Deneubourg, J.L., Franks, N., Sneyd, J., Theraulaz, G., Bonabeau, E. Camazine, Self-Organisation in Biological systems, 2001.

[2] (2011) wikipedia. [Online]. http://en.wikipedia.org/wiki/Self-organization

[3] wikipedia. [Online]. http://en.wikipedia.org/wiki/Swarm_behaviour

[4] P Angeline, "Evolutionary Optimization Versus Particle Swarm Optimization," in *Evolutionary Programming VII*, Berlin, 1998, pp. 601–610.

[5] D. Bratton and T. Blackwell, "A Simplified Recombinant PSO," *Journal of Artificial Evolution and Applications - Particle Swarms: The Second* , vol. 2008, 2008.

[6] R Brooks, "A Robust Layered Control System for a Mobile Robot," *IEEE Journal of Robotics and Automation*, vol. 2, no. 1, pp. 14-23, 1986.

[7] J. Kennedy and R. Eberhart, "Particle Swarm Optimization," in *IEEE International Conference on Neural Networks*, 1995, pp. 1942–1948.

[8] O Khatib, "Real-time Obstacle Avoidance for Manipulators and Mobile Robots," *The International Journal of Robotics Research*, vol. 5, pp. 90-98, 1986.

[9] L Gao, C Li, T Fang, and Z Xiong, "Vehicle Detection Based on Color and Edge Information," in *Image Analysis and Recognition*, 2008, pp. 142-150.

[10] F. Dellaert, D. Fox, W. Burgard, and S. Thrun, "Monte Carlo Localization for Mobile Robots," in *Robotics and IEEE International Conference on Automation*, 1999.

[11] N. Holden and A.A. Freitas, "A Hybrid Particle Swarm/Ant Colony Algorithm for the Classification of Hierarchical Biological Data," in *IEEE Swarm Intelligence Symposium*, 2005, pp. 100-107.

[12] R Poli and C Stephens, "Constrained Molecular Dynamics as a Search and Optimization Tool," in *7th European Conference on Genetic Programming*, 2004, pp. 150-161.

[13] H Shah-Hosseini, "Problem Solving by Intelligent Water Drops," in *IEEE Congress on Evolutionary Computation*, 2007, pp. 3226-3231.

[14] J.S. Vesterstrom, J. Riget, and T. Krink, "Division of Labor in Particle Swarm Optimization," in *Proceedings of the 2002 Congress on Evolutionary Computation* , Honolulu, HI., 2002, pp. 1570–1575.

[15] J. Robinson, S. Sinton, and Y. Rahmat-Samii, "Particle Swarm, Genetic Algorithm, and Their Hybrids: Optimization of a Profiled Corrugated Horn Antenna.," *IEEE Antennas and Propagation Society International Symposium*, vol. 1, pp. 314–317, 2002.

[16] G Benni and J Wang, "Swarm Intelligence," in *Proceedings of the Seventh Annual Meeting of the Robotics Society of Japan*, 1989, pp. 425-428.