International Journal of Advanced Robotic Systems

OPEN ACCESS ARTICLE

A Hybrid of Modified PSO and Local Search on a Multi-robot Search System

Regular Paper

Mohammad Naim Rastgoo[1]*, Bahareh Nakisa[1] and Mohd Zakree Ahmad Nazri[1]

1 Universiti Kebangsaan Malaysia, Bangi, Selangor, Malaysia
*Corresponding author(s) E-mail: naim.rastgoo@gmail.com

Received 23 April 2014; Accepted 13 April 2015

DOI: 10.5772/60624

Abstract

Particle swarm optimization (PSO), a new population-based algorithm, has recently been used on multi-robot systems. Although this algorithm is applied to solve many optimization problems as well as multi-robot systems, it has some drawbacks when it is applied on multi-robot search systems to find a target in a search space containing big static obstacles. One of these defects is premature convergence. This means that one of the properties of basic PSO is that when particles are spread in a search space, as time increases they tend to converge in a small area. This shortcoming is also evident on a multi-robot search system, particularly when there are big static obstacles in the search space that prevent the robots from finding the target easily; therefore, as time increases, based on this property they converge to a small area that may not contain the target and become entrapped in that area. Another shortcoming is that basic PSO cannot guarantee the global convergence of the algorithm. In other words, initially particles explore different areas, but in some cases they are not good at exploiting promising areas, which will increase the search time.

This study proposes a method based on the particle swarm optimization (PSO) technique on a multi-robot system to find a target in a search space containing big static obstacles. This method is not only able to overcome the premature convergence problem but also establishes an efficient balance between exploration and exploitation and guarantees global convergence, reducing the search time by combining with a local search method, such as A-star.

To validate the effectiveness and usefulness of algorithms, a simulation environment has been developed for conducting simulation-based experiments in different scenarios and for reporting experimental results. These experimental results have demonstrated that the proposed method is able to overcome the premature convergence problem and guarantee global convergence.

Keywords Particle swarm optimization (PSO), multi-robot search system, premature convergence problem, exploration and exploitation

1. Introduction

A behaviour-based paradigm has had a strong impact on multi-robot system research. The social characteristics of insects and animals are analysed in order to examine and apply these findings in designing multi-robot systems. There are many algorithms (e.g., GA, ACO and PSO) inspired by biological societies, which are applied in multi-robot systems to develop similar behaviours like searching [1,2,3] and foraging tasks [4]. Using a multi-robot system in searching tasks can offer several major advantages over the

single robot alternative. Searching can be run on a massive pattern in parallel. Here a significant decrease in time regarding the location of the targets and improved robustness against failure of individual robots by redundancy, as well as individual simplicity, is observed [5].

The particle swarm optimization technique is a population-based stochastic search technique introduced by [6,7]. Recently, this technique has been applied to multi-robot search systems. The first versions of PSO were proposed in [1, 2, 8] on a multi-robot search system to find a target in the environment, and studies have demonstrated that the PSO algorithm has an acceptable performance in the searching task. In several instances, adaptations of PSO have been used for multi-robot odour searches [9, 10]. Adapted versions of PSO on distributed mobile robots have been used to search the environment based on only local information [2, 11]. These adapted versions of PSO demonstrate that their performance in a group of robots is better than the basic PSO algorithm; however, this adapted version has its shortcomings, particularly when placed in an environment with a high density of obstacles.

One of the main problems of basic PSO is premature convergence; that is, the particles have a tendency to move towards the best location found and converge to that area; therefore, there is exploitative behaviour in PSO over time. It is obvious that the global searching (exploration) of PSO decreases as time progresses. Some of the variations among many that improve its performance include: fuzzy PSO [12], hybrid PSO [13], intelligent-particle swarm performance [14], addition of a queen particle [15], etc. These variations are tested on the benchmark functions.

The problem of premature convergence is also evident in the multi-robot search system in environments which contain static obstacles. In other words, one of the basic PSO properties is that the global searching or exploration of robots decreases over time and they converge to a small area, and then become unable to explore other promising areas. This problem is called premature convergence. Under this circumstance, when there are big obstacles in an environment, these static obstacles amplify this problem. These obstacles are bigger and taller than the robots and so prevent them from observing the environment behind them. When the target is placed near the obstacles, therefore, the probability of observing from the other side of the obstacles is low, and as time increases the global searching of the robots decreases. As a result they search the small area continuously, and finally converge to that small area, without being able to search the other promising areas.

Based on basic PSO, the robots' velocities in the early iterations are high due to their greater inertia weight (ω); therefore, the global searching and exploration of the robots is higher than that of local searching and exploitation. With an increase in time, the inertia weight value decreases, leading to a decrease in the global searching (exploration) of PSO. Moreover, the robots may converge to the area that may not contain the given target. A few studies are run on multi-robot search systems to solve the premature convergence problem. In [16] two new methods are proposed based on particle swarm optimization (PSO) and Darwinian particle swarm optimization (*DPSO*), named RPSO and RDPSO, respectively. These two newly developed methods are adapted to multi-robot search systems where obstacle avoidance is of high importance. The results of their studies determined that the RDPSO increases the search exploration in order to avoid being stuck in local optima and a rapid convergence to the desired objective value in comparison with RPSO.

Another problem of basic PSO is the lack of guarantee in global convergence. Establishing an efficient balance between exploration and exploitation is one of the drawbacks of basic PSO. Researchers have proposed several methods to solve this problem in different domains [17, 18]. This drawback is evident in multi-robot search systems designed on basic PSO. In a sense, in some cases, the robots are placed near the target but have to move based on the velocity and position equations of basic PSO, which may guide the robots to move to the position located farther from the target. This situation obviously causes the search time to increase, particularly when there are obstacles near the target. In this situation the best global robot (g_{best}) may be entrapped in the local optima. In [19], a new method is proposed, named MPSO, for the multi-robot search system. In their study, they added the local search method to the PSO and established a balance between exploration and exploitation. In the method, the robots are deployed in the search space and look for the target. When the robot can see the target, the value of fitness function is calculated, and if the value of fitness function in its current position exceeds half of the goal fitness function value, which is assumed by the author, the robot uses the local search strategy instead of basic PSO. In addition, the environment does not contain any big obstacles that prevent the robots from observing the area surrounding the static obstacles; therefore, there are no areas where the robot becomes stuck and after a while converges to that area.

In this article, a simple and effective PSO named the 'Modified PSO with Local Search (ML-PSO)' is proposed. This ML-PSO is based on the modification of basic PSO developed by [20], which was applied in the exploration search space. Here, a new method is proposed on a multi-robot search system which increases the global searching and guides the robots to escape from the local optima and explore different areas to find the desired target. The attempt is made to overcome two problems: premature convergence through increasing the global searching of robots, and adding a local search algorithm such as A-star to guarantee global convergence with a reduction in the search time.

The study is organized as follows. Section 2 introduces the important tools that are used by this proposed algorithm. Section 3 thoroughly describes the proposed algorithm

(ML-PSO). The computational results are provided in Section 4. Lastly, some conclusions are provided in Section 5.

2. Theoretical Background

2.1 Particle swarm optimization (PSO)

Particle swarm optimization (PSO) is a new optimization search technique which solves numerical optimization problems. In this method, particles fly through the multi-dimensional search space to find the potential solution.

To model the swarm, each particle starts to search with a randomized position (x_{id}) in the n-dimensional search space with (possibly) randomized velocity (v_{id}) (Initialization). In each step, an objective function is used to evaluate particle success (fitness function). If the value of fitness function is better than any found so far, it is stored as the best position, called p_{best}. The particle with the closest position to the goal gets the highest value in fitness function and is stored as g_{best}.

The next position vector $x_{id}(t+1)$ and the next velocity vector $v_{id}(t+1)$ of each particle are highly dependent on the current position vector $x_{id}(t)$, velocity vector $v_{id}(t)$, local best vector $p_{best}(t)$ and global best vector $g_{best}(t)$ information. Candidate solutions are optimized by flying the particles through the virtual space, with attraction to the best positions in the space indicating the best result. At each time step, the velocity is updated (next velocity) and the particles move to the new position (new position) that is calculated by the previous position and the new velocity as follows:

$$x_{id}(t+1) = v_{id}(t+1) + x_{id}(t) \tag{1}$$

The velocity of each particle is updated by the following equation:

$$\begin{aligned} v_{id}(t+1) &= \omega v_{id}(t) + c_1 * r_1 * \left(p_{best}(t) - x_{id}(t)\right) \\ &+ c_2 * r_2 * (g_{best}(t) - x_{id}(t)) \end{aligned} \tag{2}$$

Equation (2) contains three members: the first member is momentum, while the second and third members are cognitive and social components, respectively.

Momentum: $v_{id}(t)$

Cognitive component: $c_1 * r_1 * (p_{best}(t) - x_{id}(t))$

Social component: $c_2 * r_2 * (g_{best}(t) - x_{id}(t))$

Momentum pulls the particle towards the previous velocity direction. The cognitive component is the force that pulls the particle to its best position found thus far, and the social component attracts the particle towards the bestsolution found among all the particles. The inertia weight ω [21, 22] and acceleration constant c_1, c_2 are assumed to be 0.9.... 0.5and 2 and 2, respectively, and r_1, r_2 are the uniformly generated random numbers in the range of (0,1). Figure 1 shows the steps of the basic PSO algorithm mentioned above.

```
Program Basic PSO
{
   Initialization ();
   While not done
    {
          Calculate fitness function ();
          Next velocity ();
          Next position ();
      }
}
```

Figure 1. The pseudo code of basic PSO

In order to match the basic PSO to the multi-robot system, some modifications are needed. In the next section the necessary modification for this study is proposed.

2.2 Modification of PSO for the multi-robot search system

The use of one-to-one matching between particles in the PSO swarm and robots in the multi-robot system motivates our PSO-inspired multi-robot search algorithm. It is assumed that the robots have access to the map of the search space and therefore have complete knowledge about their location in the environment. If the robots do not access the whole map of the environment, it may cause a localization error, in most of these cases preventing the system from accomplishing the search task. There are some key differences between PSO and PSO in the multi-robot search that requires us to make some modifications to the algorithm.

2.2.1 Search space and static obstacles

A real search space was transformed into two-dimensional search space and discretized into non-overlapping cells. The environment in this study includes some static obstacles and a single target. The positions of the obstacles are known, but the location of the target is unknown. The cells that are occupied with the obstacles in the discretized map are marked as unsafe cells and the rest of the cells are considered as safe cells. Therefore, the robot, during the path planning, moves to the safe cells. The centre of each cell is considered a point of interest. This means that if the robot visits the centre of the cell, the entire cell is considered to be a visited cell.

2.2.2 Robot

In this study, it is assumed that the geometrical shape of the robot is like a circle with the determined radius (◎) and has the same size as a cell. The state of each robot in the search space is represented by six variables (x, y, v, θ_r, θ_c, t), which

are the position of the robot in the 2-D dimensional search space, speed of the robot, head of the robot, the direction in which the robot is determined to move to the next position, and the time in that position, respectively. The maximum turning radius is assumed to be 360 degrees; therefore, the robot can move easily to the adjacent cells around its current position (Figure 2).

As described, the search space is discretized; then the path planning of the robot from its current cell to the goal cell is discretized and the robot must cross through the centre of the cells on its route. For a single path the environment is considered to be a static world, and the problem is solved by the A-star algorithm [23]. The traditional A-star method computes the optimal path from the start position to the goal position among the static obstacles, but it fails in a dynamic environment.

2.2.3 Fitness function

It is assumed that the robot camera, which can capture a picture from the environment, is a fitness function. The robot sees up to 10 cells from its current position in eight different directions. Figure 2 shows eight directions from the current position of the robot. When the robot uses the camera to observe its surroundings, if the target is placed in the range of view of the camera (placed in its surroundings) then the fitness function based on equation (3) is calculated; otherwise, it returns to zero. The fitness function is as follows:

$$fitness\ function = \frac{\sum_{i=1}^{n} po_i}{\sum_{j=1}^{m} p_j} \tag{3}$$

Where $po_i = \{po_1, po_2, po_3 \ldots po_n\}$ a set of pixels of the target in the image is captured by the camera and $p_j = \{p_1, p_2, p_3 \ldots p_n\}$ is a set of pixels of the whole image captured by the camera. It is clear that the target is a part of the whole image; therefore, the amount of pixels of the target is less than that of the whole image, and the value of the fitness function is always (0, 1). When the robot is moving towards the target and is placed near it, the value of the fitness function is higher compared to when the robot is moving away from the target, in which case the fitness function value is decreased. If the robot cannot observe the target, the fitness function value is equal to zero, and when the robot is located exactly beside the target, this value is 1; otherwise it is between 0 and 1. The value of fitness function for each robot is calculated by them, and then these values are sent to the central station.

2.2.4 Movement limitation of robots

Although the particles in the basic PSO do not have limited acceleration and velocity, in the real world the robots have limited velocities that are discretized into discrete values. The velocity of each robot is placed between $[-v_{max}, v_{max}]$

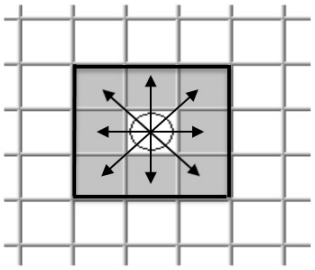

Figure 2. The adjacent cells of the robot in its current position

where the v_{max} represents the maximum velocity of the robot along its direction and the $-v_{max}$ is the maximum velocity of the robot, but in the reverse direction. If the velocity of the robot is placed out of this range, we set the maximum velocity value on each side for the robot.

2.2.5 Collision avoidance among the robots

Using the basic PSO particle displacement at each iteration, detecting any collisions that might occur along the path is not possible. Therefore, approximating the continuous movement of the robots by dividing the displacement into multiple steps and checking for collisions at each iteration is needed. In a multi-robot system, the robots and the target have some volume, and therefore have to avoid collision with each other and static obstacles. In this study, in order to avoid possible collisions among robots, the method proposed in [24] is used. In this study, it is assumed that each robot is equipped with a microcontroller that is able to calculate some simple formulae. Here, the value of the fitness function of each robot is calculated by each of them and is then sent to the central station. The next position and velocity of each robot is calculated by the central station, and then the route of each robot from its current position to the next position is generated. Finally, the collision between the path of the robots and the static obstacles in the search space is checked and the collision-free path is sent to each robot.

2.2.6 Communication among the robots

In this study, at each iteration the robots move to the next position, and the value of fitness functions in these positions are evaluated by each robot. After this, the information about their current positions and their fitness function values are sent to the central station in parallel. The central station updates the map of the environment based on the current positions of the robots. This means the current positions of the robots on the map are marked as visited cells, and are updated at each iteration. The central station based on the obtained information calculates the next velocity and next position of each robot. The next positions and next velocities of the robots are calculated based on the given algorithms (ML-PSO or basic PSO). Finally, the next positions and next velocities of each robot are sent to the

robots, and they move to them synchronously at the next iteration.

3. The algorithm

This proposed algorithm (ML-PSO) is a new version of the basic PSO that is segmented into three: 1) initialization, where each robot is placed in a random manner in the search space with the random velocity and headings. 2) The fitness function of each robot is evaluated according to Eq. (3). If the value of fitness function is greater than any found so far, it is stored as the best position, named p_{best}. The robot with the closest position to the goal gets the highest value in fitness function and is stored as g_{best}.Here, when t=0, the $p_{best}(0)$ is the first position of each robot and $g_{best}(0)$ is the first position of the first robot, so that, 3) there are two alternative strategies in guiding the robot to move to the next position (first or second). If the fitness function is larger than zero, the local search algorithm is run in order to guide the robot to move towards the target; if not, then first of all the percentage of cells visited *(PVC)*is calculated to determine whether each robot is stuck in the local optima or not. Second, the next velocity of each robot based on Eq. (5) is calculated. Finally, the next position of each robot is calculated according to basic PSO, Eq. (1). Steps 2 and 3 above are repeated until the termination criteria are met.

The termination criteria are essential for obtaining the proper solution in a rational time. In this study, if one of the robots reaches the target or the number of iterations exceeds the maximum iterations, which are assumed to be 400 iterations, the termination criteria are reached and the program terminates.

If the number of iterations exceeds 400 iterations, it means that the algorithm could not find the target.

Figure 3 represents the steps of ML-PSO.

Two strategies are proposed in this method. The first strategy establishes an efficient balance between exploration and exploitation by adding the local search method. The second strategy attempts to overcome the premature convergence problem by attracting the robots to the unrevealed area and increases global searching.

3.1 First strategy: Local search

The basic PSO cannot guarantee the global convergence of the algorithm. In the ML-PSO, the local search algorithm can guarantee that the robot reaches the target when the robot is placed near it, and its fitness function is more than zero. The reason behind this strategy is that the local search is better able to guide the robot towards the target when the robot is near it, while the basic PSO may guide the robot to escape from the target. This means that when the robot becomes closer to the target, it will be able to track the target faster by applying local searches. In this study, A-star algorithm [23] is applied as a local search algorithm.

```
Program ML-PSO
{
   Initialization ();
   While the termination criteria are not met
    {
          Calculate fitness function ();
          If Fitness Function>0
                                  //first strategy
                    Local-search ();
          Else
          {
                                  //second Strategy
                   Calculate PVC ();
                   Update velocity ();
                   New position ();
          }
      }
}
```

Figure 3. The pseudo code of ML-PSO algorithm

If heuristic function never overestimates the actual solution cost, then A-star always finds an optimal solution if one exists, which is the main advantage of A-star in comparison with other heuristic algorithms. Another advantage of A-star is that it considers fewer nodes than any other admissible search algorithm with the same heuristic. These properties make this algorithm the most well-known heuristic search algorithm that is widely used in path finding methods to determine the local search [25].

In this study, when the fitness function of each robot is larger than zero, its next position is calculated based on the local search strategy (A-star) instead of the second strategy. In the A-star, the search is started from the current node, and it continues until reaching the determined look ahead. The look ahead is defined as the number of next positions that the robot has to move to. In this study, the look ahead value is equal to 1. As described before, there are eight adjacent cells around the current position of the robot that it can move to in a specific direction. When the camera of the robot rotates, it can evaluate the fitness function for each of eight directions. Then the A-star algorithm, by selecting the highest f-value that belongs to the specific direction, moves towards the adjacent cell along this specific direction. The f-value for these directions is calculated by the following equation:

$$f(n) = g(n) + h(n) \tag{4}$$

According to Eq. (4), the h (n) is the cost-to-go, which is assumed to be the fitness function value of the robot's current position in the specific direction. Meanwhile, g (n) is the cost-thus-far, which is the cost from the current node to the next position, and as the look ahead is equal to one, the g (n) in this study is equal to one. A-star uses two lists: Open and Close. The Open list is the list that stores all the

acceptable directions of the robot which has a specific fitness function value, and then sorts them. The sorting of the Open list is based on the Max-Heap in this study, and as each direction is added to the Open list, the list is reordered based on the highest f-value. This means that the top of the list corresponds to the highest f-value. The selected direction with the highest f-value pops up from the Open list and is put into the Close list. The algorithm selects the direction from the Close list and then calculates the next position of the robot based on the selected direction, a position located among the adjacent cells around its current position. Finally, the robot moves to the next position.

3.2 Second strategy: Overcome the premature convergence problem

When the fitness function of each robot returns to zero, it means the robot did not see the target in this iteration, and as time increases based on the basic PSO property its global searching decreases, and the local searching increases. Therefore, the probability of becoming entrapped in a small area (local optima) increases and robots may converge to the small area, meaning they cannot search the other areas to find the target.

The ML-PSO is inspired by the modification of PSO introduced by [20]. The measurement of fitness function in this method is different. The authors assume that each robot can sense its surroundings and calculate how much space around it is unrevealed. This means that when the robot is placed in one cell, by accessing the map of the search space it can mark its position as a visited area and then calculate how much of the space around its current position it has not yet visited. In order to increase the global searching, the cognitive component (p_{best}) is considered a dynamic component. This means the p_{best} in each iteration represents the position with the highest fitness function among all particles. Here, a fourth component is added to the velocity equation to increase the global searching. The velocity equations are represented in equation (5):

$$\begin{aligned} v_{i,j}(t+1) &= \omega.v_{i,j}(t) \\ &+c_1.r_1.\left(p_{best}(t)-x_{i,j}(t)\right) \\ &+c_2.r_2.(g_{best}(t)-x_{i,j}(t))+c_3.r_3.(p_{cd}-x_{i,j}(t)) \end{aligned} \quad (5)$$

In ML-PSO, the four presented components that affect the movement of the robots are similar to those in [20]: momentum, cognitive and social, and the fourth component which is the force between robots and unexplored space. The fourth component in the velocity equation represents the vector p_{cd} [20], assumed to be directed towards the most distant spot of the unexplored areas, and is modified based on this domain. In [20], the value of c_3 is considered a constant, but when applied in this domain it does not lead to the desired solution. In this study, the value of c_3 is considered variable and changes based on a *PVC* variable, which is described in detail in due course. Here, the movement observed is not necessarily directed towards the most distant spots, because of the variable nature of *PVC*, which affects the c_3 generating variable vectors.

In [20], though, the global searching of basic PSO is increased, and this method overcomes the premature convergence problem; however, it has some drawbacks when applied to the searching domain. One of the drawbacks of this method is that when the target is placed near the robots and the robot cannot visit it in the first iteration, it is attracted through the fourth component (p_{cd} vector) to immediately search the other unexplored areas. These robots are not able to search the area surroundings them due to the presence of the p_{cd} vector with a fixed size and dynamic p_{best}. In other words, the size of p_{cd} vector in this method is considered a constant, and attains a high value because of c_3 ; therefore, the robots have to leave their current areas immediately and move towards the furthest away unexplored areas when this component is added. Although the presence of dynamic p_{best} increases the global searching, the robot loses its best experience achieved so far; hence, the local searching of the algorithm is not considered.

In this proposed method (ML-PSO), the cognitive component (p_{best}) for each robot in the first iteration (t=0) represents their current positions, and in the next iterations will be updated when the fitness function value of the robots' current positions becomes higher than the p_{best}.

In order to increase the global searching features of ML-PSO, when the robot is stuck in a small area (local optima), as well as providing a chance for the robots to search their surroundings when they arrive at a new area, the vector size of the fourth component is considered variable by assigning a variable value to c_3.

Whether or not the robots are trapped in the local optima is determined by their observation of their surroundings and the calculation of how much of their surrounding area is explored. Equation (6) defines this phenomenon:

$$PVC = \frac{visited\ cells\ in\ the\ surrounding\ area}{all\ cells\ in\ the\ surrounding\ area} \times 100 \quad (6)$$

In this study, the value of c_3 is changed based on the *PVC* variable. There exists a direct relation between the *PVC* value and the assigned value of c_3 ; therefore, if the vector size of the fourth component is low, the robot has to search its surroundings. Consequently, a high value of *PVC* makes the c_3 value higher, which directly causes the vector size of the fourth component to become larger.

In order to assign an appropriate value to c_3 the *PVC* is calculated first. Based on Eq. (6) at each iteration, each robot calculates the amount of cells visited in its surroundings so far; if this value is high then it means that this area has been

searched before by other robots and the robots could not observe the target in this area yet. In other words, based on the Basic PSO formula (Eq. (2)), as time increases the value of ω decrease and the global searching (the exploration) behaviour of PSO decrease; therefore, robot has become entrapped in this small area (local optima) and is searching its surroundings continuously. As a result, when the value of *PVC* exceeds 50%, it means that it is possible that the robot may become entrapped in the local optima.

In order to guide the robots to escape from that area, they have to search the other unexplored areas. By increasing the value of c_3 based on increasing the value of *PVC*, the robots can escape from that region and pull towards the other unexplored areas (Table 1). When the value of *PVC* is less than 40%, the value of c_3 is set to zero, and the fourth component is ignored. In this case the robots move and search their surroundings based on the basic PSO velocity and position equations.

PVC	Value of c_3
$PVC < 40\%$	Zero
$40\% < PVC \leq 50\%$	$\frac{1}{2}c_1$
$50\% < PVC \leq 60\%$	c_1
$60\% < PVC \leq 70\%$	$2c_1$
$70\% < PVC \leq 80\%$	$3c_1$
$80\% < PVC \leq 90\%$	$4c_1$
$90\% < PVC \leq 100\%$	$5c_1$

Table 1. The value of c_3 based on the value of *PVC*

4. ML-PSO result

4.1 Simulation conditions

The ML-PSO and basic PSO are simulated and tested in three different environments with four different target positions in 100 test cases. The three environments with increasing numbers of obstacles are simulated and presented as (Env. 1, Env. 2, Env. 3) in order to show the function and the performance of this proposed algorithm (ML-PSO) when exposed to different numbers of obstacles and the premature convergence problem. As discussed before, one of the main problems of basic PSO is the premature convergence which appears in this domain when there are obstacles in the environment. Therefore, the presence of obstacles in this environment is essential and contributes towards obtaining an acceptable result. The four target positions are placed near or behind the obstacles to simulate the worst cases. In addition, the initial positions of the robots are located in the worst places but in a random manner; this means the initial positions of the robots are placed randomly at the farthest positions from the target, and they cannot see the target in the first iteration. The four points of the target in Env. 3 with maximum number of obstacles are shown in Figure 4. Only one target position is activated during each search run.

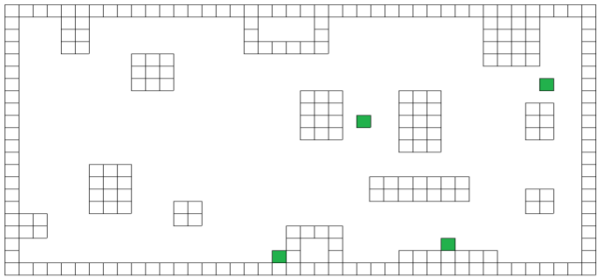

Figure 4. The map of simulation search space (Env. 3) and the four different target point locations

Unlike most of the PSO simulations, the search space in this simulation is bounded. Due to the condition approximation of the robot's actual searching, the search space in this simulation has a hard border. It is assumed that if the next position of each robot were placed out of the search space, it would reverse in order be placed inside the search space.

For the simulation results, the following parameter values are used:inertia coefficient, ω is set to 0.9.... 0.5 [21] and the c_1, c_2 coefficients are both set to 2, 2. An initial v for each robot is set to simulate the behaviour of the physical robot. In this study only three robots are used, and therefore the *lbest* method is the same as the *gbest*.

Here, the basic PSO algorithm is adapted to the multi-robot search system; therefore, unlike most of the PSO research studies that track function value, this simulation searches the target function. The simulation stops when the robot reaches the target or the maximum number of iterations (400 iterations) has elapsed.

4.2 Simulation result

To evaluate the effectiveness of ML-PSO, there are two scenarios:

1. In order to show and compare the performance of ML-PSO and basic PSO when exposed to the premature convergence (local optima) problem, the amount of space explored by the robots in three different environments with an increasing number of obstacles is evaluated. These three environments contain six, 10 and 14 obstacles, respectively.

2. The search time is selected as a measurement to compare the performance of ML-PSO and basic PSO algorithms when exposed to the two mentioned problems. In this scenario the search time consumed by ML-PSO and basic PSO algorithms in three environments with an increasing number of obstacles is evaluated as 100 runs.

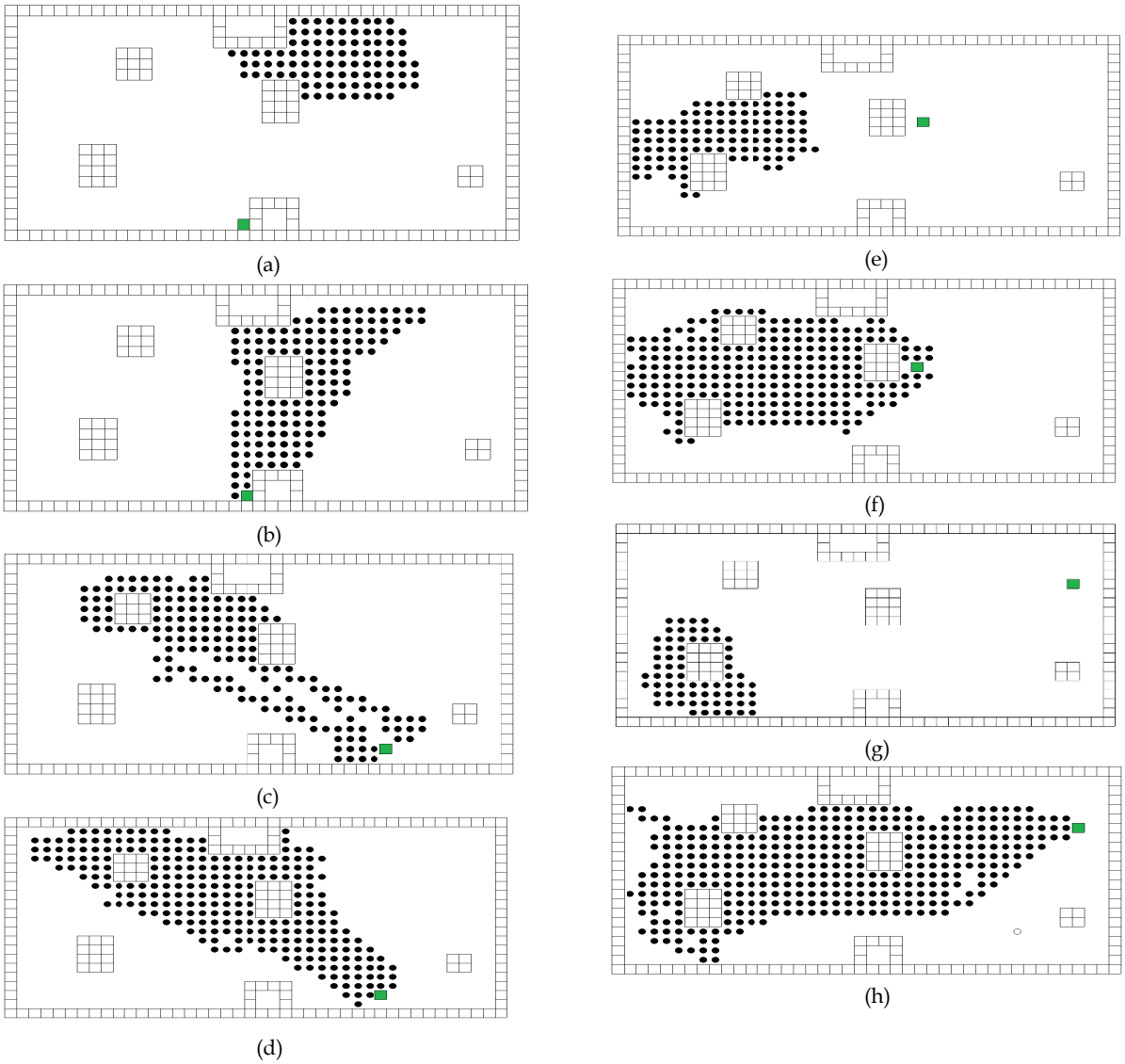

Figure 5. The results of simulated ML-PSO and basic PSO algorithms in Environment 1 with six static obstacles in 100 runs. The explored space is represented by black dots and the target is shown by the green square. (a) The simulation result of the basic PSO algorithm with target point 1. (b) The simulation result of ML-PSO with target point 1. (c) The simulation result of basic PSO algorithm with target point 2. (d) The simulation result of ML-PSO with target point 2. (e) The simulation result of basic PSO algorithm with target point 3. (f) The simulation result of ML-PSO with target point 3. (g) The simulation result of basic PSO algorithm with target point 4. (h) The simulation result of ML-PSO with target point 4.

The number of iterations in this article is considered to be 400 based on trial and error, because in most cases if the robots cannot reach the target before 400 iterations it means they are stuck in the local optima and never reach the target in this proposed domain. Since the main objective here is to decrease the search time, 400 iterations could be considered a proper number.

4.2.1 Simulation result in Environment 1

The areas explored by three robots in four different target positions in the environment (Env. 1) which include six obstacles are shown in Figure 5(a-h). Here, it is assumed that the minimum number of obstacles is six, which prevents the robot seeing behind the obstacles.

The number of areas explored by applying ML-PSO and basic PSO on a multi-robot search system in this environment with six obstacles is evaluated.

Here, Figure 5(a) and 5(b) show the areas explored by basic PSO and ML-PSO algorithms, respectively, at the target point 1. In this case, the basic PSO could not find the target and the robots are stuck in the local optima; therefore, the area explored by basic PSO is smaller. The ML-PSO in this case can find the target and reach it. In order for ML-PSO to reach the target it explores more areas, which is not true for basic PSO.

As seen in Figure 5(c) and 5(d), the robots subject to basic PSO and ML-PSO can find the desired target and reach it. In this situation, though the initial positions of the robots

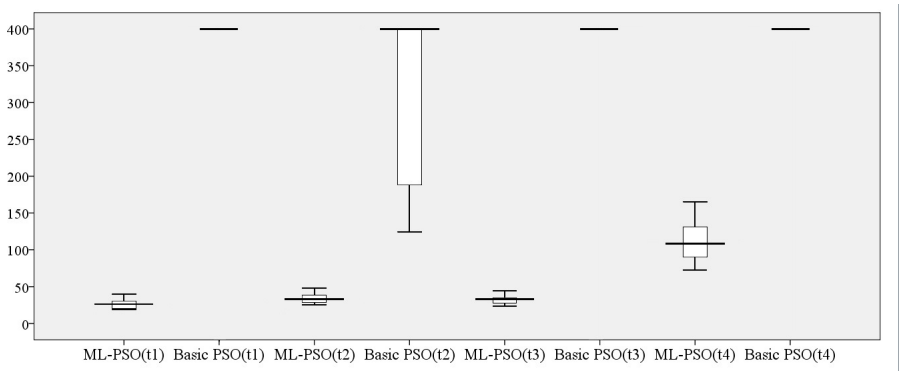

Figure 6. The result of the search time of the ML-PSO and basic PSO algorithm with four different target positions in Environment 1 containing six static obstacles in 100 runs

are located in the worst positions in relation to the target, the small number of obstacles in the way of the robots allow the observation of the target in the first iterations. Although the behaviour of ML-PSO in this situation is similar to that of the basic PSO, more unexplored areas are explored through this proposed method during 100 runs.

When the target is placed behind the obstacle and the initial position of the robots is placed as far as possible from the target (4(e)), the robots could not find the target and reach it during 100 test cases through a basic PSO algorithm. The reason here is that the initial positions of the robots are located in the worst positions and are entrapped in the local optima. In this case, in early iterations, when the global searching of the robots is high, none of the robots see the target due to the presence of the obstacles and the limitations of observation by the robot camera. Here, the robots can see only 10 cells around themselves, so in the first iterations they could not observe the target. As the number of iterations increases, the global searching of the robots reduces, and they converge to small areas; hence, they become entrapped in the local optima and never reach the target.

As seen in Figure 5(f), although the target is placed behind the obstacles, ML-PSO helps the robots to escape from the local optima and search more unexplored areas during 100 runs by applying the second strategy. Then, by adopting the local search method (first strategy), the robots move towards the target more quickly.

The target point 4 is located near the corner. This case is very difficult for basic PSO when the initial position of the robots is placed in the farthest position, where the robots could be entrapped in the local optima and converge to that area, while in the case of ML-PSO, when each of the robots finds that it is stuck in the local optima it tries to leave that area and move towards the unexplored areas by applying the second strategy.

Another strategy to evaluate the performance of both of the algorithms in this environment is to compare the number of iterations passed by each algorithm in finding the target. The search times consumed by ML-PSO and basic PSO algorithms in this environment with the four different target positions are compared in Figure 6.

The basic PSO algorithm could find the target in one of the target points (target point 2) but it could not find the target in target points 1, 3 and 4 during 400 iterations in 100 runs (Figure 6). The ML-PSO algorithm can find the target in four different target points in a reasonable search time.

As seen in target point 1, the basic PSO algorithm could not find the target in a given time during 100 runs. This is because the robots are stuck in the local optima and cannot escape. The number of iterations for the ML-PSO algorithm in this case ranges between 20 and 50 iterations.

The number of iterations in ML-PSO and basic PSO slightly increase in target point 2. The figure shows that the basic PSO algorithm can find and reach the target in between 130 and 400 iterations while the ML-PSO can find and reach the target in between 30 and 50 iterations.

In the remaining number of target points (target points 3 and 4), basic PSO could not find the target due to the premature convergence problem. That is, the robots converge to the small area and never search other areas.

As observed, in these two difficult target positions the ML-PSO can find the target. The function of this algorithm is to guide the robots to escape from the local optima and move towards the unexplored areas through determining whether or not they are stuck in the local optima. Here, when each robot observes the target, it moves towards it using the first strategy. The number of iterations of ML-PSO in target point 3 varies between 25 and 50 iterations. The search time in target point 4 in ML-PSO increases and reaches around 70-140 iterations.

4.2.2 Simulation result in Environment 2

The areas explored by the robots in four different target positions in the environment (Env. 2) with 10 obstacles in 100 test cases are shown in Figure 7(a-h). This environment is more complex than the previous environment, and we therefore expect the weakness of basic PSO to be more evident than before.

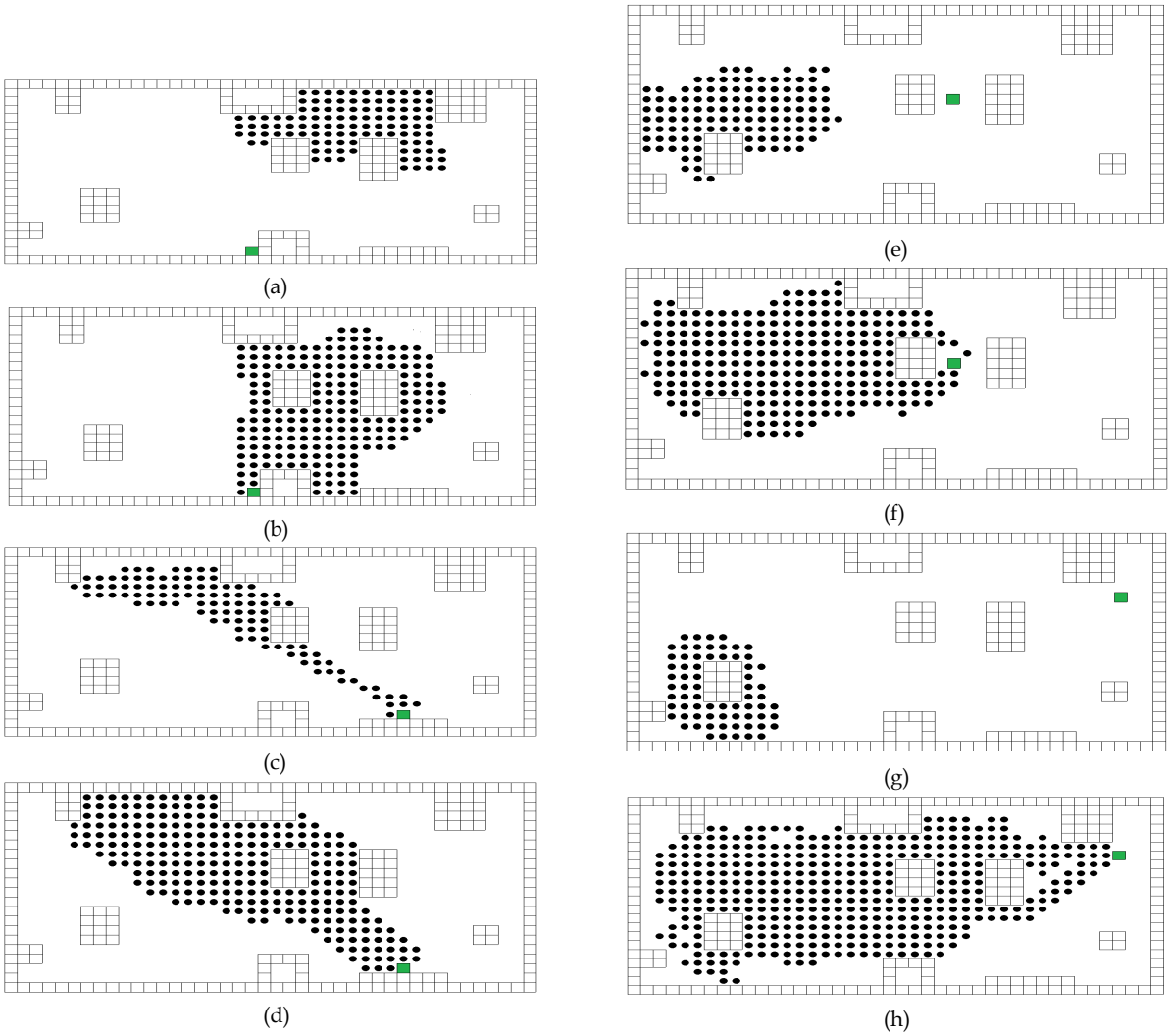

Figure 7. The results of simulated ML-PSO and basic PSO algorithms in Environment 2 with 10 static obstacles in 100 runs. Explored space is represented by the black dots and the target is shown by the green square. (a) The simulation result of basic PSO algorithm with target point 1. (b) The simulation result of ML-PSO with target point 1. (c) The simulation result of basic PSO algorithm with target point 2. (d) The simulation result of ML-PSO with target point 2. (e) The simulation result of basic PSO algorithm with target point 3. (f) The simulation result of ML-PSO with target point 3. (g) The simulation result of basic PSO algorithm with target point 4. (h) The simulation result of ML-PSO with target point 4.

The performance of the basic PSO and ML-PSO in exploring the environment with 10 obstacles with four different target positions in 100 runs is shown in Figure 7(a-h). This environment (Env. 2) is more complex than Env. 1. When the number of obstacles increases, the probability of observing the target by the robots decreases.

As seen in Figures 7(a-h), the basic PSO algorithm can only find the target and reach it at the target point 2 (6(c)) because the position of obstacles in this environment is similar to the previous environment (Env. 1) with the same target point. In this case, the added obstacles are not placed in the way of the robots, and thus they can find and reach the target.

In the remaining target points (6(a), 6(e), 6(g)), the basic PSO algorithm fails to find and reach the target. The multi-robot search system using the basic PSO algorithm is stuck in the local optima and converges to that area; therefore, they cannot search other unexplored areas. This is because of the number of static obstacles and the initial positions of the robots which are closer to the target.

On the other hand, ML-PSO algorithm in four different target points can find the target and reach it. Figure 7(b) represents the amount of explored areas with ML-PSO algorithms. The number of explored areas with ML-PSO is more than the number forprevious environments with the same target position. This is because of the obstacles being added to the environment in a way that diverts the robots to search the obstacles' surroundings more extensively.

As shown in Figure 7(c) and 7(d), although the basic PSO could also find the target in this case and reach it, the amount of areas explored by the ML-PSO is more than with basic PSO. In this case, the robots' movement through the

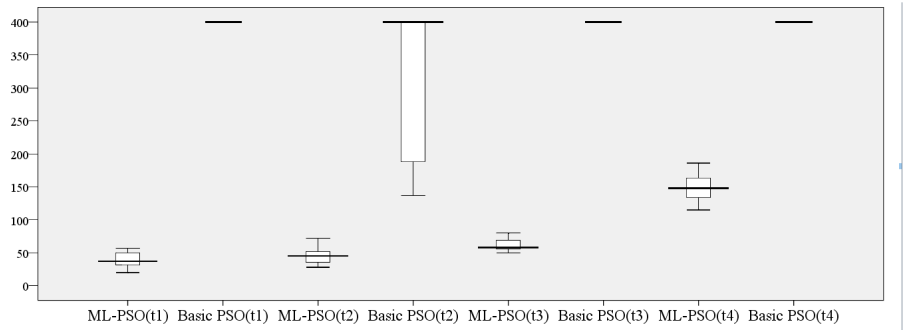

Figure 8. The result of the search time of ML-PSO and basic PSO algorithm with four different target positions in Environment 2 containing 10 static obstacles in 100 runs

basic PSO towards the target is the same as through ML-PSO, but they search the same areas during the search time because the global search decreases over time.

Figures 7(e) and 7(f) represent the amount of explored areas using the basic PSO and the ML-PSO when the target is placed in a difficult position (behind the obstacle). In this case, the robots must move to the nearby area to see the target. Global searching in the basic PSO decreases over time, and thus the robots are stuck in the local optima and cannot explore other areas. In the ML-PSO algorithm, global searching (exploration) is increased by applying the second strategy. This strategy pulls the robots to move towards the unexplored areas; hence, they observe the target by searching different areas. The amount of exploration by the ML-PSO is about twice that of the basic PSO.

The difficult case in this environment is target point 4. In this case the target position is near the corner of the search space and the initial position of the robots is very far from the target; therefore, the robots converge to local optima and search that area continuously, and the amount of areas explored though the basic PSO is low, while the robots can find and reach the target through the ML-PSO algorithm. In contrast, the amount of explored areas when applying ML-PSO algorithm on a multi-robot search system is high due to the initial position of the robots, which is far away from the target, and this algorithm has to maintain the global search at a high level to search many areas and find the target.

Another strategy to evaluate the search time of ML-PSO and basic PSO in this environment is illustrated in Figure 8. Figure 8 represents the search time (number of iterations) the ML-PSO and the basic PSO needed to find the target.

As can be seen in Figure 8, in all target points except one the basic PSO algorithm fails to find the target and cannot reach it. The basic PSO in target point 2 can find the target and reach it, and this case is similar to Environment 1 (Env. 1) with the same target point.

The number of iterations for ML-PSO in this environment increases due to the added static obstacles. The number of iterations in the target point 1 for ML-PSO is about 30-60 iterations, while these figures increased in target point 2 and reached about 40-70 iterations. When the target is placed behind the obstacles (target point 3) it is difficult for the robots to find it; therefore, the search time increases and ranges from 50 to around 100 iterations.

In target point 4, the target is placed in one of the hardest positions and the multi-robot is located at the farthest position from the target. This position shows the weakness of the basic PSO in finding objects among 10 obstacles. In this case, a multi-robot search system applying basic PSO could not find the target, but ML-PSO can find the target during 130-170 iterations despite the fact that this case is very difficult.

4.2.3 Simulation result in Environment 3

This environment is the most complex environment in this study, and contains 14 obstacles. The amount of explored areaswith both basic PSO and ML-PSO is shown in Figure 9(a-h).

Figure 9(a-h) shows the amount of explored areas through the ML-PSO and the basic PSO algorithms in the environment (Env. 3), which are the most complex environments in this study and contain 14 static obstacles with four different target positions in 100 test cases.

As shown in Figures 9(a),9(c), 9(e) and 9(g), the basic PSO algorithm failed to find the target in four different target points and could not reach the desired target during 100 runs. In these target positions, the multi-robot search system is stuck in local optima and searches the same areas during the desired search time. As a result, the amount of explored areas is low.

On the other hand, although the environment is complex, ML-PSO can successfully find the target in all target positions and reach it in a given search time. Figures 9(b) and 9(d) show the amounts of explored areasusing ML-PSO algorithms. These amounts are higher than in other environments (Env. 1, Env. 2) with the same target points.

When the target is in the hardest place, such as target points 3 and 4 (behind the obstacle and near the corner), the ML-PSO algorithm guides the robots to search unexplored

Figure 9. The results of simulated ML-PSO and basic PSO algorithms in Environment 3 with 14 static obstacles in 100 runs. Explored space is represented by the black dots and the target is shown by the green square. (a) The simulation result of basic PSO algorithm with target point 1. (b) The simulation result of ML-PSO with target point 1. (c) The simulation result of basic PSO algorithm with target point 2. (d) The simulation result of ML-PSO with target point 2. (e) The simulation result of basic PSO algorithm with target point 3. (f) The simulation result of ML-PSO with target point 3. (g) The simulation result of basic PSO algorithm with target point 4. (h) The simulation result of ML-PSO with target point 4.

areas further by increasing global searching; therefore, they move towards the target and then move faster by using the local search strategy.

The search time of the ML-PSO and the basic PSO in this environment with four target positions is shown in Figure 10.

Figure 10 represents the search time consumed through the ML-PSO and the basic PSO algorithms in the most complex environments in this study. As can be seen in Figure 10, through ML-PSO the robots can find the target in four different target points, which is not true for the basic PSO algorithm. The number of iterations in four target points increases in the ML-PSO algorithm in comparison with previous environments (Env. 1, Env. 2).

In target point 1, the ML-PSO spends around 35-100 iterations while the basic PSO fails to find the target, and the multi-robot system is stuck in the local optima. As in previous environments with the same situation can be seen, the basic PSO algorithm failed to find the target and became entrapped in the local optima. When the global searching of the algorithm is at a high level, the static obstacles in the way of the robots prevent them from observing the target

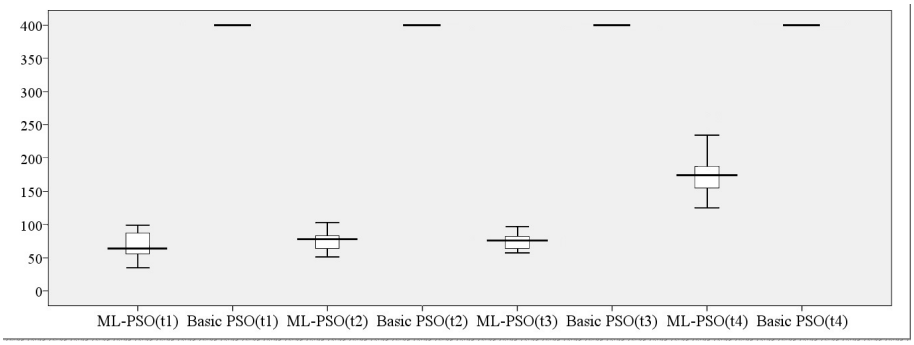

Figure 10. The results of the search time of ML-PSO and basic PSO algorithm with four different target positions in Environment 3 containing 14 static obstacles in 100 test cases

in the first iterations; hence, they become entrapped in the local optima, and as time increases they search the same areas and cannot find the target.

In target point 2, the number of iterations required for ML-PSO to find the target is between 60 and around 130. Although in the previous environments with the same target point the basic PSO could find the target and reach it, in this environment the additional static obstacles in the way of the robots lead to the robots being stuck in the local optima and never finding the target, while the ML-PSO can find the target and reach it in between 70 and 100 iterations.

The number of iterations in the basic PSO when the target is placed in target point 3 is 400. This means the robots could not find the target through the basic PSO algorithm in this case. In this case, ML-PSO algorithm can find the target and reach it, and the number of iterations passed by this algorithm varies from 70 to 100.

Target point 4 is the most difficult among all cases in all three environments. In this case, the target is placed near the corner and in the most distant possible position in the search space. The basic PSO quickly became entrapped in local optima and could not search the other areas to find the target, while the ML-PSO could find the target and reach it in 130-250 iterations.

5. Conclusion

A biologically inspired search strategy is developed and tested for robot swarms. This search technique, named ML-PSO, is based on bird flocking and basic PSO. The ML-PSO algorithm is able to solve two problems of basic PSO, taking into account static obstacles in the search space (Figure 3 represents the pseudo code of the ML-PSO algorithm). The first problem of basic PSO is premature convergence, which appeared in this domain when there are static obstacles in the search space and the initial positions of the robots are far from the target. In basic PSO, as time progresses, the global searching (exploration) of the robots decreases and the robots converge to a small area; thus, the system fails to search the other areas and find the target. In order to solve this problem, ML-PSO increases the global searching of the basic PSO by adding the fourth component to the velocity equation (second strategy), which is inspired by [20] and modified based on this domain. The value of the fourth component varies in different cases when each of the robots detects whether it is stuck in the local optima or not. This variable helps the robot to escape from the local optima and move towards and search the unexplored areas.

Another problem is that basic PSO cannot guarantee that the global optima (the target) will be reached and cannot establish a balance between exploration and exploitation. This means that in some cases when the robots see the target and the fitness function value is high enough, the basic PSO algorithm guides the robot to move to the positions that may increase the distance between the robot and the target. In order to decrease the search time and guarantee that the robots can reach the target when they observe it, the local search method is added to the ML-PSO algorithm. When the robot sees the target, instead of using modified basic PSO equations (second strategy), it moves towards the target by applying a local search method such as A-star (first strategy).

The ML-PSO and basic PSO algorithms are simulated and tested on the multi-robot search system in the three environments, with increasing numbers of static obstacles with four different target positions. The results show that the ML-PSO algorithm has better performance in comparison with the basic PSO, as observed, and the amount of areas explored through ML-PSO is higher than through basic PSO in the same situation. In addition, the search time in ML-PSO is less than in basic PSO in all three environments with four target positions.

This proposed method could be applied in real-world environments with the same situations. This method can be used to solve problems in finding an object in environments with many static obstacles,e.g., factories, hazardous environments, detecting water and oxygen in moon exploration projects, etc.

It is worth mentioning that during the simulations in this study one of the possible drawbacks could be the handling of the localization task with respect to modification in the

search environment. For instance, adding moving objects into the search space while their movement information is unknown to the robots could lead to the failure of the robots in their localization task. Therefore, more in-depth studies are necessary to overcome this drawback.

6. Acknowledgements

The authors are indebted to the anonymous referees for their comments and suggestions. They helped us to significantly improve the paper. The research reported here was supported by the Ministry of Education Malaysia Fundamental Research Grant Scheme (FRGS/2/2014/ICT02/UKM/02/1).

7. References

[1] Doctor, S., Venayagamoorthy, G. K., & Gudise, V. G. Optimal PSO for collective robotic search applications. Paper presented at Congress on Evolutionary Computation, CEC 2004.

[2] Pugh, J., Segapelli, L., & Martinoli, A. Applying aspects of multi-robot search to particle swarm optimization. International Workshop on Ant Colony Optimization and Swarm Intelligence, Brussels, Belgium, 2006, pp. 506-507.

[3] Das, A., Kantor, G., Kumar, V., Pereira, G., Peterson, R., Rus, D., Singh, S., & Spletzer, J. Distributed search and rescue with robot and sensor teams. To appear in Field and Service Robotics, 2003, Japan.

[4] Di Chio, C., Poli, R., & Di Chio, P. Extending the particle swarm algorithm to model animal foraging behaviour. International Workshop on Ant Colony Optimization and Swarm Intelligence, Brussels, Belgium, 2006, pp. 514-515.

[5] Sahin, E. Swarm robotics: From sources of inspiration to domains of application. In Swarm Robotics: State of-the-Art Survey Ser. Lecture Notes in Computer Science (LNCS 3342), E. Sahin & W. Spears, Eds. Berlin Heidelberg: Springer-Verlag, 2005, pp. 10–20.

[6] Eberhart, R., & Kennedy, J. A new optimizer using particle swarm theory. Proceedings of the Sixth International Symposium on Micro Machine and Human Science, 1995, MHS'95.

[7] Kennedy, J., & Eberhart, R. Particle swarm optimization. Proceedings of IEEE International Conference on Neural Networks, 1995.

[8] Hereford, J. M. A distributed particle swarm optimization algorithm for swarm robotic applications. Study presented at the IEEE Congress on Evolutionary Computation, CEC 2006.

[9] Jatmiko, W., Sekiyama, K., & Fukuda, T. A PSO-based mobile sensor network for odor source localization in dynamic environment: Theory, simulation and measurement. Study presented at the IEEE Congress on Evolutionary Computation, 2006, CEC.

[10] Marques, L., Nunes, U., & de Almeida, A.T. Particle swarm-based olfactory guided search. Autonomous Robots, 2006, 20(3), 277-287.

[11] Pugh, J., & Martinoli, A. Inspiring and modeling multi-robot search with particle swarm optimization. Study presented at the IEEE Swarm Intelligence Symposium, 2007, SIS 2007.

[12] Shi, Y., & Eberhart, R.C. Fuzzy adaptive particle swarm optimization. Proceedings of the 2001 Congress on Evolutionary Computation, 2001.

[13] Lovbjerg, M., Rasmussen, T.K., & Krink, T. Hybrid particle swarm optimiser with breeding and subpopulations. Proceedings of the Genetic and Evolutionary Computation Conference, 2001.

[14] Ciuprina, G., Ioan, D., & Munteanu, I. Use of intelligent-particle swarm optimization in electromagnetics. IEEE Transactions on Magnetics, 2002, 38(2), 1037-1040.

[15] Clerc, M. The swarm and the queen: Towards a deterministic and adaptive particle swarm optimization. Proceedings of the 1999 Congress on Evolutionary Computation, 1999, CEC 99.

[16] Couceiro, M.S., Rocha, R.P., & Ferreira, N.M.A novel multi-robot exploration approach based on particle swarm optimization algorithms. Study presented at the 2011 IEEE International Symposium on Safety, Security, and Rescue Robotics (SSRR), 2011.

[17] Angeline, P. J. Evolutionary optimization versus particle swarm optimization: Philosophy and performance differences. Paper presented at Evolutionary Programming VII, 1998.

[18] Vesterstrom, J. S., Riget, J., & Krink, T. Division of labor in particle swarm optimisation. Proceedings of the 2002 Congress on Evolutionary Computation, 2002, CEC'02.

[19] Darvishzadeh, A., & Bhanu, B. Distributed multi-robot search in the real-world using modified particle swarm optimization. GECCO (Companion) 2014, 169-170.

[20] Masár, M., & Zelenka, J. Modification of PSO algorithm for the purpose of space exploration. Study presented at the IEEE 10th International Symposium on Applied Machine Intelligence and Informatics (SAMI), 2012.

[21] Xin, J., Chen, G., & Hai, Y. A particle swarm optimizer with multi-stage linearly-decreasing inertia weight. International Joint Conference Computational Sciences and Optimization, CSO 2009, vol. 1, 505–508. IEEE, 2009.

[22] Shi, Y., & Eberhart, R. A modified particle swarm optimizer. IEEE International Conference on

Evolutionary Computation, 1998 Anchorage, Alaska.

[23] Hart, P. E., Nilsson, N. J., & Raphael, B. A formal basis for the heuristic determination of minimum cost paths. IEEE Transactions on Systems Science and Cybernetics, 1968, 4(2), 100-107.

[24] Liu, F., Narayanan, A., & Bai, Q. Effective methods for generating collision free paths for multiple robots based on collision type. Proceedings of the 11th International Conference on Autonomous Agents and Multi-agent Systems, 2012, vol. 3.

[25] Korf, R. E. Real-time heuristic search. Artificial Intelligence; 1990, 42(2): 189-211.