An Performance Analysis of a PSO-Based Algorithm for Swarm Robotics

Rodrigo Melo
E-mail: rodrigo.melo89@yahoo.com.br
Clodomir J. Santana Junior
E-mail: cjjsj@ecomp.poli.br
Caio Victor E. Albuquerque
E-mail: cvea@ecomp.poli.br
Sergio Campello Oliveira
E-mail: scampello@ecomp.poli.br
Polytechnic School of Pernambuco, University of Pernambuco
Recife, Pernambuco, Brazil

ABSTRACT

The development of an algorithm responsible for coordinating a group of autonomous robots is a challenging research topic, which is frequently addressed by researchers in Swarm Robotics field. Hence, Swarm Robotics and Swarm Intelligence share a great number of principles. Therefore, it is common to adapt algorithms from the swarm intelligence to coordinate swarms of robots. The Particle Swarm Optimization is a well-known swarm algorithm, which can be adapted to coordinate a swarm of robots. In this work, the performance of a PSO-based algorithm was assessed in two different maps. The first map simulates an open area with obstacles, while the second map has the characteristics of an apartment. To analyze the impact of the fitness function in the algorithm performance, three different fitness functions were used. The results achieved in both maps are not only satisfactory, but reveal the importance of the fitness function to the swarm performance.

Keywords: Particle Swarm Optimization, PSO, Swarm of Robots, Swarm Robotics, Braitenberg

1. INTRODUCTION

Swarm robotics (SR) is a field of great research interest. One of the reasons is the large number of applications. There are several approaches regarding the swarm coordination. The most usual way is to use a technique based on an algorithm of artificial swarm intelligence, such as PSO, ABC, ACO, BFO and others [1], [2].

This paper presents the development and assessment of a PSO-based algorithm for swarm robotics, which has to explore and search for targets in an unknown environment. The targets are static and do not emit any kind of signal. Furthermore, the only information about the environment is its size, which is 25 square meters.

In order to verify if the swarm is able to adapt to different scenarios, the maps depicted in Figure 1 were developed using V-REP (Virtual Robot Experimentation Platform) [3]. They represent, respectively, an open area with obstacles (map 1) and an indoor environment such as an apartment (map 2). Another characteristic of the maps is the presence of access points. In both scenarios, there are only two access points whose surrounding area is used to initialize the robots initial position.

The authors thank the Polytechnic School of Pernambuco, CAPES, FACEPE and CNPq for partial financial support for this paper.

Fig. 1. (a) - Open area with obstacles map (map 1). (b) - Apartment map (map 2).

The aim is the simulation real cases, for instance, the search for an explosive in a room where the robots could only enter through one or two entrance.

Given the importance of the fitness function to the algorithm performance, three fitness functions were adopted in this work. To assess these functions, two metrics were used:

1) Coverage Area: the ratio between the area explored by the robots and the map total area.

2) Success Rate: the percentage of targets found by the swarm.

The rest of this paper is organized as follows: A brief introduction to PSO-based algorithms applied to Swarm Robotics is given in Section II. In Section III, all the modifications required to adapt the canonical PSO algorithm are reported. The experiments are described on Section IV. The results and conclusion are presented in Sections V and VI, respectively.

2. PARTICLE SWARM OPTIMIZATION ALGORITHM IN SWARM ROBOTICS

Depending on the characteristics of the problem, the use of SR is preferable than a single robot. The reason lies in the intrinsic characteristics of swarms, with relatively simple individuals, that are able to perform complex tasks. In fact, the main characteristic of a swarm of robots is the scalability, flexibility and robustness [1].

The robustness, simplicity and low computational cost are aspects which made the PSO an attractive optimization
method for SR applications [5, 6] and [11]. However, different approaches were made towards fitness functions and obstacles avoidance.

Several modifications were already proposed to create a PSO-based algorithm for robot swarms [5], [8], [9], [10], [11], and [12]. Nevertheless, none of them assessed the impact of different fitness function on the swarm performance nor the performance in different environments.

There are several ways to model the objective function of the problem. On one hand, papers such as [5], [8], [9], and [12], use targets that emit a signal (song, heat, smell, etc.). In this case, the fitness function is the intensity of the signal received. On the other hand, in [11], the objective function is defined as the amount of unexplored area that has been explored in the last $T$ seconds.

In PSO, as in other swarm algorithms, the fitness function is the element that provides the individual a feedback of how good the last movement was. Therefore, it is important to study the impact that different fitness functions could have in the works mentioned before.

In addition to the fitness function examination, the environmental analysis is also important when assessing algorithms for swarm of robots. The characteristics of the swarm algorithm make it more appropriate for specific problems or search spaces. There are problems that have a large number of local minima/maxima, requiring algorithm with high exploration capability. Similarly, algorithms for swarm of robots can be sensible to environment changes, so it is important to verify its performance in different scenarios to identify the set of characteristics which the algorithm is indicated.

3. PROPOSED PSO-BASED ALGORITHM FOR ROBOTS COORDINATION

In this work, we made simple adaptations on the PSO algorithm to make it more appropriated to this type of application. Among the modifications made, a mechanism to avoid obstacles and collisions with other robots can be cited.

In addition to the collisions avoidance mechanism, the communication between robots was also adapted. In the vanilla PSO, there is no physical limitation to the communication range of the particles in the search space. However, when dealing with swarm of robots this is not always true. Depending on the communication technology adopted, i.e. Bluetooth, Wi-Fi, radio, etc., there is a maximum communication range associated. In this paper, the robots use simulated wireless technology to communicate. Therefore, the robots communicate using broadcast with every robot inside its communication range. In other words, the set of neighbors of a robot is defined as all the other robots which are within the robot’s communication radius. Using this protocol, robots can request and send information of their pBest, which is the best position found by the robot itself, and fitness values if needed.

The collision avoidance mechanism works as follows: First, the velocity is calculated using the PSO velocity equation (Eq. (1)).

$$v_{i}^{t+1} = \omega v_{i}^{t} + c_{1} r_{1}(pBest_{i}^{t} - x_{i}^{t}) + c_{2} r_{2}(lBest_{i} - x_{i}^{t}) \quad (1)$$

Where $v_{i}^{t+1}$ is the new velocity of the particle $i$ in the iteration $t$, $\omega$ is the inertia coefficient, $r_{1}$ and $r_{2}$ are random numbers generated using a uniform distribution between [0,1], $c_{1}$ and $c_{2}$ are the cognitive and social coefficient respectively. Finally, pBest is the best position found by the particle $i$ and lBest is the best position found by a neighborhood.

After the new velocity is calculated, the particle position is updated according to Eq. (2).

$$x_{i}^{t+1} = x_{i}^{t} + v_{i}^{t+1} \quad (2)$$

Where $x_{i}^{t+1}$ is the new position of particle $i$.

Next, using the proximity sensors, the robot verify the presence of obstacles, such as walls and other robots, in the path that takes the robot to the calculated position. If any obstruction is detected, a Braitenberg vehicle model, similar to the one used in [13], is adopted. However, instead of the odor source, the position generated by the PSO is used in this work. This led to the motion of contouring the obstacles while trying to follow a similar path to the one calculated by the PSO. Otherwise, a forward kinematics approach is used to move the robot to the position generated by the PSO.

The flow chart in Figure 2 shows how the movement calculation described above is calculated.

During the execution of previous experiments, a few issues were noted, but due to time constraints they moved away from the scope of this work. First, no adaptation was made toward maintain the swarm cohesive. The reason is that the objective is to explore most of the maps area, and if the robot keeps trying to maintain aggregation, the area explored by the swarm will be limited. Second, the localization of the robots was abstracted since it is not in the scope of this article. For this reason, it was used a function that returns the position (similar to an ideal GPS) of the robot.

Each robot in the swarm executes the steps described in Algorithm 1 which is the algorithm used in this work.

4. EXPERIMENTS

To determine the appropriate number of iterations, which was used as stop criteria, previous experiments were conducted. It was noticed that after 1000 iterations, the swarm was able to achieve around 30 percent of coverage area. However, it had not converged or were stuck in some point of the map. In fact, the swarm was still exploring and could achieve better results.

Fig. 2. Flow chart of the movement calculation.
Algorithm 1 PSO-Based algorithm for Swarm Robotics

1: Go to initial position in the access points;
2: Set pbest and lbest as current position;
3: while stop criterion is not met do
4:  Verify the presence of targets nearby;
5:  Evaluate current position;
6:  Communicate with the robots in the neighborhood;
7:  Update pbest and lbest
8:  Update velocity using Equation 1
9:  Calculates new position applying Equation 2
10: Move the robot following the flow chart (Figure 2)
11: end while

with a higher number of iterations. Therefore, the number of iterations were increased to 1400. Although even better results could be achieved with more iterations, due to time constraints this number was not increased.

The simulations were created and executed using V-REP PRO EDU Platform [3], which is an environment for modeling, programing and executing simulations with robots. Both maps have 25 square meters of area, with walls and static targets. In the apartment map (Map 2), Figure 1 (b), the doors also represent another component/obstacle. Some of them can be automatically open using a proximity sensor when a robot get close to them and others are locked. Due to the locked doors, marked with a red rectangle on Figure 1 (b), only 88% of the map can be accessed and 70% of the targets can be found. Moreover, the door marked with a green rectangle opens outwards, blocking the rest of the corridor thus creating an area difficult to visit. This hard access area contains 13% of the map and one target. Thus, only 75% of the environment has the appropriated accessibility and 60% of the targets are located in those areas. The motivation to create a map with inaccessible (closed doors, blocked paths, etc.) areas is to simulate real cases, where the swarm has to locate those areas in order to inform the base. So another robot, more robust, can be sent to clear the path.

The targets used in this research do not emit any kind of signal that can be detected by the robots from distance. However, since the targets are the only red object in the scenarios, the robots can use a color sensor or a camera to identify the targets. As a result, the robot is not able to detect targets located behind obstacles (door, walls, etc.). For simulation purpose, the detection is made by the recognition of the target’s color. When a target is detected by one robot, this information is stored and the robot keeps searching for other targets. This behavior increases both the coverage area and the probability of finding more targets.

A total of 20 scenarios were executed in each environment, and in each fitness function tested. The scenarios differ in the number of robots in the swarm, or in the number of targets displayed on the map. In some cases, both will vary. The size of the swarms will be 5, 10, 15 and 20 robots. While the number of targets 2, 4, 6, 8 and 10.

The fitness functions used in this article were:

1) Standard (stand): The fitness function is the Euclidean distance to the closest target. This function has the objective to minimize the distance to the closest target. In real applications, this function could be used to find routes to known targets positions into an unknown map, as in disaster sites.

2) Random target (rand_tar): It follows the same idea as the standard, using the Euclidean distance to the targets and, like the last one, the objective is to minimize the result. However, the robots do not know the targets positions, so, it generates temporary random targets until the sensors detect a real target.

3) Distance from base (dist_base): This function calculates the Euclidean distance to the base (launch point). Here, the objective is to minimize the inverse of this distance.

The values of the PSO parameters are presented in the Table 1. The communication range and the obstacles detection range values were determined after previous experiments, and the others parameters use standard values proposed in literature [14], [15].

<p>| TABLE I EXECUTIONS PARAMETERS. |</p>
<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cognitive Coefficient (c1)</td>
<td>2.05</td>
</tr>
<tr>
<td>Social Coefficient (c2)</td>
<td>2.05</td>
</tr>
<tr>
<td>Inertia weight (ω)</td>
<td>0.8</td>
</tr>
<tr>
<td>Communication range</td>
<td>1.0m</td>
</tr>
<tr>
<td>Obstacle detection radius</td>
<td>0.5m</td>
</tr>
</tbody>
</table>

5. RESULTS

As suggested in Section 4, since the number of iterations was adopted as stop criteria, it represents a constraint to the swarm performance. As we increased the number in 40% (from a thousand to a thousand and four hundred), it was possible to see a significant improvement in coverage area. Table II compares the average coverage area achieved by each function with 1000 and 1400 iterations on Map 1.

Even though it was observed a great improvement on coverage area numbers, even better results could be achieved if a higher iteration number was used. This is due the fact that the swarm was instructed to continuously search for targets. Then, when a target is found by a robot, this information is registered and the robot keeps exploring the environment, searching for other possible targets. Hence, the swarm exploration is a continuous process, which can be interrupted only by the limit of iterations.

In Figure 3, 5 and 7, the results of coverage area for all functions in map 1 can be seen. In Figure 4, 6 and 8, the results of coverage area in the map 2 is shown. Finally, the success rates results for 20 robots for each fitness function used on the open area and apartment map can be seen in the Figure 9 and 10.

In all the cases studied, the apartment map was proved harder to explore than the open map. Even when just the accessible

| TABLE II BEST COVERAGE COMPARISON MAP 1. |
|------------------|------------------|------------------|------------------|
| Fitness Function | Best Coverage area for 1000 iteration | Best Coverage area for 1400 iteration | Number of Robots |
| stand           | ≈ 22.5%          | ≈ 65.0%          | 20               |
| rand_tar        | ≈ 28.5%          | ≈ 79.0%          | 20               |
| dist_base       | ≈ 26.5%          | ≈ 74.0%          | 20               |
area is considered, only in rand_tar the results are similar in both maps. It was also verified that the coverage area increases as the number of robots increase, as expected.

In the stand function (Figure 3 and 4), the number of targets will influence the coverage area because the robots are attracted to the targets. However, even when the number of targets increases, the robots may explore less because they are attracted for the closest targets. Therefore, when the number of targets increases, some targets will appear closer to the initial swarm position.

The rand_tar function showed to be promising. The results achieved for coverage area were the best ones among the studied functions, reaching nearly 80% in the open map (Figure 5) and 60% in the apartment map (Figure 6). However, if only the accessible area is considered, that value will be of 68%. Moreover, if the 12% of hardly accessed area are neglected, this value will go up to 80%, which is the same as the one achieved in the Map 1 (Figure 1) (a).

The dist_base function showed reasonable results in the open map (Figure 7), reaching a median of 75% of coverage area when using 20 robots. However, the coverage area in the apartment map was unsatisfactory (Figure 8), especially when compared to rand_tar. This could be due to the fact that in this map, the robots have a considerably limited space to move and, since they do not have a movement coordination strategy, the robots were unable to explore the map freely.

In terms of success rate, all functions have similar results. It is worth mentioning how the targets are added in the Map 2 (Figure 1) (b) because some of them are inaccessible. Table III shows this distribution. Low accessible areas are parts of the map, which require an additional maneuverability effort to the
The fitness function stand showed a slight decrease in the success rate when there was an increase in the number of targets in the second map (Figure 10). Although, when there are 6, 8 and 10 targets, the same number of targets is found which is 3. For 2 and 4 targets, it is noticeable the advantage of this function compared with the other two, a reason for that is the fact that the robots in the standard function know the position of the targets. On the open map, the stand function is stable, finding half of the targets in most cases. The only exception is when there are 8 targets, which the median is close to 40%.

The dist_base and rand_tar showed similar results to the one achieved with stand (Figure 9 and 10). Moreover, when the cases with 8 and 10 targets are analyzed, the rand_tar achieved better results and dist_base achieved, basically, the same result as the stand function. This can be explained with the fact that, as mentioned before, the stand function acts as a rout discover, since the robots know the targets position but need to discover a path to them. This makes the success rate increase without the necessity of map exploration. Moreover, it is important to remember that in both maps there are targets in low accessibility areas. In fact, in the apartment map, some targets are not accessible.

### 6. Conclusions

The results presented in this work indicate that the functions proposed are flexible and can be used in different maps, such as the open area and apartment map. As analyzed in previous tests and confirmed with the experiments, the swarm could achieve better results of coverage area if the number of iterations were increased. It has to be studied if increasing beyond 1400 iterations will increase the coverage area. Moreover, the rand_tar and the dist_base functions were able to achieve similar and, in some cases, even better results than the stand function. This is important because in unknown environments the targets locations are uncertain and this makes the standard function not applicable to these experiments while the other two functions do not have this limitation.

Regarding the apartment map, even though the results achieved were satisfactory There are still, possibilities for improvement, especially in the dist_base function. When compared to the open area map, the apartment map has narrow path (or halls) and since...

<table>
<thead>
<tr>
<th>Total number of Targets</th>
<th>Number of inaccessible targets</th>
<th>Number of targets in low accessible areas</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>4</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>6</td>
<td>2</td>
<td>0</td>
</tr>
<tr>
<td>8</td>
<td>2</td>
<td>0</td>
</tr>
<tr>
<td>10</td>
<td>3</td>
<td>1</td>
</tr>
</tbody>
</table>
the swarm does not have mechanisms to coordinate the robots movements in this situation, it could represent a bottleneck that prevents the robots from reaching better results.

The function random target was the best solution presented in this paper, reaching 80% and 40% of coverage area and success rate, respectively, in the open map, and 60% and 50% of coverage area and success rate, respectively, in the apartment map.

This paper indicates the importance of fitness functions in the performance of a swarm robotics. As future work, other fitness functions can be proposed to analyze the performance in order to find an even better solution. The development of a movement coordination strategy for environments with narrow halls or low accessibility areas can also be proposed. In addition to that, other evaluation metrics could be included and the PSO-based algorithm used in this work could be compared with others swarm robotics algorithms, which can be based on other swarm intelligence techniques and using different concepts, such as the idea of using a sink node.

7. References