Democratic Inspired Particle Swarm Optimization for Multi-Robot Exploration Task

Oussama Moslah, Yassine Hachaïchi, Younes Lahbib

Democratic Inspired Particle Swarm Optimization for Multi-Robot Exploration Task

Oussama MOSLAH[a,*], Yassine HACHAÏCHI[b], Younes LAHBIB[c]

[a] *Université de Tunis El Manar, Ecole Nationale d'Ingnieurs de Tunis, LR11ES20 Laboratoire d'Analyse, de Conception et de Commande des Systèmes (LACS), 1002, Tunis, Tunisie*
[b] *LAMSIN-ENIT, Université de Tunis El Manar, TUNISIA.*
[c] *Electronics and Micro-electronics Laboratory, Université de Monastir, TUNISIA.*

Abstract

In this paper, we propose a new method for exploring an unknown environment with a team of homogeneous mobile robots. The goal of our approach is to minimize the exploration time. The challenge in multi-robot exploration is how to develop distributed algorithm to govern the colony of robots while choosing its new direction so that they simultaneously explore different regions. In this paper we use the extended version of Particle Swarm Optimization (PSO) to robotic applications, which is referred to as Robotic Particle Swarm Optimization (RPSO), a technique to compute robots new location. To better adapt this technique to the collective exploration problem, and maximize the exploring area, we propose a new method for computing PSO's global best parameter. Experiment results obtained in a simulated environment show that our new method of computing PSO's global best parameter increases the explored area with a shorter time convergence.

Keywords: Swarm robotic, Multi-robot system, mobile robotics, collective exploration, SLAM, Particle Swarm Optimisation.

1. Introduction

The exploration and mapping of an unknown environment, which is also defined as simultanius localization and mapping (SLAM), is a fundamental problem for mobile robots and multi-robot systems. Many applications such as rescue [1, 2] mowing [3] and cleaning [4] require full coverage of the environment,its center of gravity, and also the sides, in the fastest time possible (especially in rescue operations).

Explore an environment using several robots instead of one is much more advantageous. But in order to avoid the exploration of the same surface multiple times, or by more than one robot, there is a need to develop a movement strategy which allows the robots to coordinate their movements, and without the use of a centralized control unit.

In the collective exploration of an unknown space, each robot must: maintain communication with other robots to share the progress and the discovered surface over a period T, detect and distinguish between obstacles and free space, avoid obstacles when moving, and choose the best path where a robot can move to discover a maximum space that hasnt been discovered yet.

Particle Swarm Optimization (PSO) is one of the techniques used for collective exploration of swarm robotic systems which abides by those requirements. PSO is an optimization technique developed by James Kennedy and Russell Eberhart in 1995 [5] that models a set of potential solutions to a particle moving in the space search. The aim of this method is to be suitable for multi-robot applications, improvement has been made to take into account the real world environmental constraints.

In this paper, in order to develop our strategy to govern the whole colony of heterogeneous robots, we propose a method inspired from Particle Swarm Optimization. We adapt it to the exploration task. Our idea is to control the robots in order to explore the space on two steps: in the first, they try to converge to their center of gravity to explore it, which is probably the center of the space to explore. While in the second step, robots will carry on to explore the sides. This way, we will have the map of the entire space.

With the aim of exploring the space in two steps as explained above, we have changed the way of calculating the global best solution of the PSO (it will be detailed in the section 4) in order to calculate it by taking into account all local optima of the colony, not only the best performing candidate.

This paper is organized as follows. In section 2, we will discuss related works. Section 3 presents the theoretical background of the PSO and its application in the multi-robot field RPSO, in section 4 we will introduce our

*Corresponding author
Email addresses: oussama.moslah@enit.rnu.tn (Oussama MOSLAH), Yassine.Hachaichi@ipeiem.rnu.tn (Yassine HACHAÏCHI), Younes.Lahbib@enicarthage.rnu.tn (Younes LAHBIB)

theoretic approach to help robots decision in motion coordinated exploration with mobile robots. Section 5 presents series of experiments which are carried out in simulations and results. Finally, we discuss the results and use cases in Section 6.

2. Related works

The multi-robotic exploration, instead of one, is nowadays an advantageous approach which has several known implementation techniques. These techniques can be classified into three categories: techniques in which the robots move randomly, techniques that require planning the movements of robots in advance, and techniques in which robots do not plan their trips in advance, but they coordinate with each other.

Techniques where robots move randomly: In order to explore an unknown environment, the author in [6] focuses on the relative location between robots and random robots exploration. The goal is to generate a collective map of the explored area. The system works as follows. Initially, robots might not know their relative positions. In such a case, each robot explores on its own map. The explored portion increases in the robots memory until two robots can communicate. In this case, they begin exchanging data and estimating their relative location. Once they have a good hypothesis for their relative location, they actively verify this hypothesis using a rendezvous technique. If successful, the robots form an exploration map, they combine their data in a shared map and begin to coordinate their exploration activities. On the other hand, if the assumption of relative location proves to be false, robots continue to explore independently and exchange data in order to refine their estimations of relative location. During the exploration, the size of the explored map increases more and more when a new robot can determine its relative position.

Another approach was presented in [7], in which authors represent the environment as a matrix of pixels (occupancy grid maps). They define discovered pixels that have undiscovered adjacent pixels as frontier pixels. After detecting frontier pixels, the algorithm considers the possibility to send a robot to a pixel by calculating the cost of moving the robot to each pixel and estimating the utility of the pixel. The cost of moving is proportional to the distance between the robot and the pixel. On the other hand, the utility of the pixel depends on the number of robots that move towards that pixel or at a nearby location.

Keeping in mind that the robots moves randomly, algorithms based on one of those techniques, will takes more time especially when the map is quite complex and big compared with the number of robot. Contrarily with our approach, in which the robots movement is coordinated, enables us to optimize the time of exploration.

Techniques that require planning of the robots movement in advance: Generally divide the environment into regions, and affect a robot to every region in order to explore all regions. One of those methods, allowing the division of the environment to regions, is the Voronoi diagram [8]. It is a distribution of the map to regions based on the distance between a point and a reference. The reference can be a robot or an obstacle, it depends on the application. In each region, each point is closer to the robot (or obstacle) in which the region is affected than others.

In reference [9] authors propose a method for the detection and trapping of an object / target using algorithms inspired by the bacterial chemotaxis (BC). They followed four main steps: the first step is the establishment of a coordination system between robots [10], this step consists of choosing a robot of the colony as a location reference. Then, the positions of the other robots (or any point in the area) will be determined based on their distances from the reference robot. The second step is to build the Voronoi diagram [8]. Step 3 is the target research phase (victim, terror ...). In this step each robot is looking for the target in its region using the algorithm imitating BC. In step 4 robots surround targets detected in step 3 using the same algorithm (BC). In order to solve the problem of distributed control of the colony, each robot moves in the search space by imitating the movement mechanism of BCs. While moving, a bacterium seeks to find a maximum concentration of nutrient. The main steps of the BC algorithm are:

- Put to the target position a large concentration of nutrient.
- Calculate the new direction of the bacteria.
- The bacterium moves in its direction previously calculated in order to go forward to the target.
- The nutrient concentration in a pixel decreases whenever the bacterium finds it.

Similar work has been introduced in [11] in which authors presented their approach to the inspection of an environment which know its map previously. The algorithm is summarized in the following steps:

- Define regions to be inspected by the robot using the Voronoi diagram.
- Identify the frontiers cells to be traveled by the robots.
- Calculating the cost of moving each robot to each frontier cell.
- Associate each robot to a frontier cells using the Hungarian method [12].
- Finally, repeat the assignment of robots to frontiers until travel up all frontiers.

In [13] the authors also divided the map into regions, but they divided it into equal rectangular segments. At first they affect robots to segments randomly. When two robots

meet, they exchange research results. Then each robot changes its direction to the least explored segment.

Opposed to our exploration approach of an unknown environment, these approaches require prior knowledge of the research area so that they can divide it into segments. In addition, the decomposition of the field is done at a single computer (or central control or leader robot). Therefore, the malfunction of the calculator will lead to the stopping of all robots. Also, robots are not synchronized as in our approach. In other hand, robot's motion doesn't depend on status/position of other robots, so, the robot's motion, while exploring, isn't fully coordinated.

Techniques where robots do not plan their trips in advance: As in our approach, robots recalculate their new direction in each iteration based on the surroundings and other robots. Using these methods, robots seem more coordinated than using others previously presented.

In [14] authors implemented an extended version of Particle Swarm Optimization (PSO) [5] which is an algorithm inspired by the movement of birds in search of food. This version is called Robotic Darwin Particle Swarm Optimization (RDPSO). In which, authors have implemented two key ideas in PSO:

1. Adapt PSO to robotic applications RPSO, in which they added two vectors, the first vector allows the robot to avoid obstacles, and the second vector allows maintaining communication between all the robots.
2. In the second idea, they used the Darwinian punishment and reward mechanism (DPSO). It divides the colony in sub-colonies, the algorithm punished the sub-colony that is not improved by subtracting the least performing robot, and rewards sub-colony which improves by adding a robot from the excluded ones. This mechanism allows the colony to avoid being trapped in sub-optimal solutions.

Other works inspired by the behavior of social insects and animals have been compared in [15]. Authors have compared five algorithms implemented in the problem of exploration. In their article, they compared the RDPSO with the Extended PSO (EPSO) [16] and Physically-embedded PSO (PPSO) [17], which are three algorithms inspired from the PSO. And they compared them with Glowworm Swarm Optimization (GSO) [18] and Aggregation of Foraging Swarm (AFS) [19].

EPSO is one of the first extended versions of PSO adapted to the constraints of the real world. The main idea in the EPSO is that each robot only considers the robots within the limits of its communication area. After calculating the motion vector, authors use the Braitenberg [20] obstacle avoidance algorithm.

In PPSO, like RDPSO, there is no central agent for coordinating the movements or actions of robots. The algorithm also requires full synchronization of robots. Robots can calculate their position only after all other robots exchange needed information (eg, individual solutions). Also, a robot does not share its position if its own solution isnt the best solution throughout the colony. This reduces communication traffic. However, it is also necessary that the robots are stopped after each iteration to process all relevant information. In addition, the collision avoidance behavior was not considered in the equation of the algorithm. Unlike the approach that we adopted where avoidance collusion and obstacle are taken into account when calculating the displacement. Instead of that, when a robot will be trapped (or gets stuck), it moves back and turns right.

GSO is an algorithm inspired by the behavior of Glowworm. Where each robot (Glowworm) chooses a nearby robot and goes toward him. The robot which has the greater brightness value (luciferin) has the highest probability of being chosen as the robot to what its neighbors are heading. The brightness value is completely proportional to the relevance of the solution found by the robot. This will lead to divide the colony of robots into sub-colonies. Each colony has a sub-leader. All robots in the same sub-colony will diverge to the same point/solution (to their leader), and explores in his neighborhood. The problem here is that all the robots of the same sub-colony will explore the same surface. This is the same problem using RDPSO. Maybe in other tasks, such as finding food, will be a good idea to split the colony into sub-colonies. Unlike our approach, which is more suited to the exploration problem. The authors also used a low obstacle avoidance mechanism.

In AFS, robots move following attractive and repulsive forces between robots. The force of attraction dominates over long distances, and the force of repulsion dominates over short distances. In this approach, the authors consider obstacles as a part of the objective function of the colony. In other words, if the robots need to maximize a given measure, for example, find the highest density of victims in a disaster site, the obstacles are considered global minima of their objective function.

The results of comparison made in [15] show a better result in terms of percentage exploration using RDPSO in any configuration (number of robot, communication distance). On the other side, in the problem of collective exploration, the global optimum notion is not completely true. In fact, a site that has been explored at time t is not a solution for the time t + 1. So the use of the Darwinian principle has no great influence on the operation of the algorithm.

Our approach extends the Robotic Particle Swarm Optimization idea in order to generate a faster exploration algorithm. In contrast to [6] and [7] in our method, robots dont move in the search space randomly, but they choose their new directions in order to move to the less discovered area. Also, in contrast to [9], [11] and [13], our approach doesnt require any prerequisites information about the map shape or size. This makes our approach more flexible and easier to implement in all shapes of maps. However, similar to [14], [16], [17], [18] and [19], algorithms

using our approach dont need any prerequisites knowledge of the area, and the decision making is fully decentralized. But comparison results presented in [15] shows a better result found using RDPSO presented in [14]. In this paper we will present an extended version of PSO that democratizes the colony while choosing their new directions. Experimental results show a faster exploration using our method.

3. PSO and RPSO: Theoretical background

This section is organized as follows: first we will introduce the PSO technique, and then we will discuss its main variants, after that we will introduce its extended version to the multi-robot applications as presented in [14].

3.1. Particle Swarm Optimization

Particle Swarm Optimization is a stochastic method inspired by le behavior of the flocking of the bird, it has been developed by Dr. Kennedy and Dr. Eberhart in 1995 [5]. The system is initialized with a population of random particles. Each particle is a potential solution, moves in the search space in order to find the optimal solution. A particle is characterized by its position (Pos) and its objective function (OF), and which moves into the search space by following:

- The local optimal solution in its knowledge, minimizing or maximizing its objective function;
- The global optimal solution, which is the best solution between local optima of the whole colony;
- Without completely changing its direction.

In every optimization step, each particle has to update its new solution through equations 1 and 2.

$$\begin{aligned}\Delta Pos(t) = \quad & w.\Delta Pos(t-1) \\ & +r_l.w_l.(Pos_{lBest} - Pos(t-1)) \\ & +r_g.w_g.(Pos_{gBest} - Pos(t-1))\end{aligned} \tag{1}$$

$$Pos(t) = Pos(t-1) + \Delta Pos(t) \tag{2}$$

Wherein $Pos(t)$ is the desired position, $\Delta Pos(t)$ the velocity of the particle, Pos_{lBest} the optimal solution in its local knowledge (Local Best), Pos_{gBest} the global optimal solution (Global Best) in the swarm knowledge (thats doesnt mean the best solution can be found), coefficient w, w_l and w_g are the assigned weights to the inertia influence, local optimum solution and global swarm optimum solution respectively, t is the time (or iteration number) r_l and r_g are random vectors between 0 and 1.

PSO is a computing method, which allows iteratively the growth of the results of potential solution in order to optimize a problem. The evaluation of the convergence of the optimization problem to a solution is through an objective function which reflects the result of the potential solution.

The idea of our approach is to suggest a new method to calculate the position of the global optimum Pos_{gBest}, which is more adapted to the collective exploration problem of multi-robot system, a detailed explanation of the classic method (classic Robotic PSO) and our new method (Democratic Robotic PSO) will be presented in the following sections.

3.2. Particle swarm optimizations variant and extended version

Since the first appearance of the PSO, and due to its effectiveness and simplicity in implementation, it has been applied in many optimization problems such as power system [21], Robotics [22] [23], transportation [24, 25], data clustering [26] and electronics [27]. This has led to the development of many variant in order to adapt it in deferent field.

In [28] authors reviewed diferent studies focusing on PSO, advantages and disadvantages, the basic variants, modifications and applications that have implemented using PSO. They divided the deferent versions of PSO into two groups: Basic Variants, and Modifications. Basic Variants are influenced by the PSO control parameters such as dimension of the search space, number of particles, acceleration coefficients. The modification in PSO consists of three categories: extension of field searching space, adjustment the parameters, and hybrid with another technique.

In this paper, we will adjust the global best parameter, in order to adapt the extended version of PSO, called RPSO, to the multi-robot exploration problem.

3.3. Robotic Particle Swarm Optimization

PSO technique has been extended to multi robot system application [14]and referred to as Robotic Particle Swarm Optimization (RPSO), in addition to the PSO parameters, RPSO takes into account real-world constraints to avoid obstacles and keep robots at a convenient distance to maintain communication between them without interfering with each other.

In every optimization step, each robot has to update its new position through equations 3 and 2.

$$\begin{aligned}\Delta Pos(t) = \quad & w.\Delta Pos(t-1) \\ & +r_l.w_l.(Pos_{lBest} - Pos(t-1)) \\ & +r_g.w_g.(Pos_{gBest} - Pos(t-1)) \\ & +r_{Obs}.w_{Obs}.(Pos_{Obs} - Pos(t-1)) \\ & +r_{Comm}.w_{Comm}.(Pos_{Comm} - Pos(t-1))\end{aligned} \tag{3}$$

Wherein Pos_{obs} the optimal solution that allows the robot to avoid obstacles, Pos_{Comm} is the optimal solution that permits the robot to keep communication with other robots within the swarm, coefficient w_{obs} and w_{Comm} are the assigned weights to the avoidance obstacle component and maintaining communication component, r_{obs} and r_{Comm} are random vectors between 0 and 1.

An illustration of the influence of the parameters in the final velocity of the robot can be seen in Figure 1.In which the robot is represented by a blue triangle, a half circle represents its limit of detection, the optimal solution is shown in green and an obstacle is schematized as black rectangle.

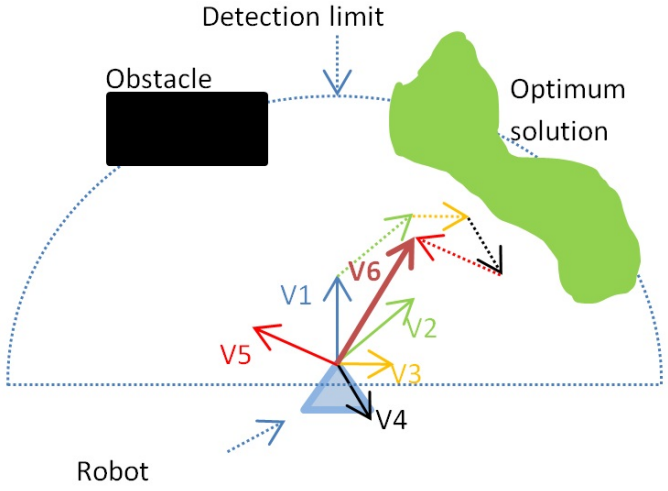

Figure 1: Robot position's updating

Vectors V1, V2, V3, V4, V5 and V6 shown in Figure1 represent respectively :

- V1 which is $w.\Delta Pos(t-1)$, represents the old velocity at time $t-1$ robot. Weighted by w to increase or decrease the importance of this vector. This vector is used to keep the robot loyal to its previously made choices.
- V2 which is $r_l.w_l.(Pos_{lBest} - Pos(t-1))$, represents the cognitive vector. Weighted by w_l which is used to increase or decrease the importance of this vector, and a random number r_l which give it a stochastic aspects. This vector is used to drive the robot toward to the position maximizing its objective function from its own local knowledge.
- V3 which is $r_g.w_g.(Pos_{gBest} - Pos(t-1))$, represents the global knowledge vector. Similar to V2, it is weighted by w_g and a random number r_g. This vector is used to drive the robot toward to the position maximizing its objective function from global robots knowledge.
- V4 which is $r_{obs}.w_{obs}.(Pos_{obs} - Pos(t-1))$, represents the avoidance obstacle vector. Similar to V2, it is weighted by w_{obs} and a random number r_{obs}. This vector is used to drive the robot far from obstacle based on its own local sensing.
- V5 which is $r_{Comm}.w_{Comm}.(Pos_{Comm} - Pos(t-1))$, represents the maintaining communication vector. Similar to V2, it is weighted by w_{comm} and a random number r_{comm}. This vector is used to keep the robot at convenient distance in order to maintain communication between all robots within the swarm.
- V6 which is $\Delta Pos(t)$, represents the new robot's velocity. It's computed by the of V1, V2, V3, V4 and V5.

In the following section, we present our proposed modification in the RPSO's global best solution in order to help robots to chose it in a democratic way.

4. Democratic Robotic Particle Swarm Optimization

In order to adapt the RPSO to the multi-robot exploration problem, in this section, we propose a new method which calculates the position of the global best solution in democratic way.

Even if democratizing the swarm isn't a new idea, the way how democratizing it makes the difference. In [29] authors introduced Democracy-inspired Particle Swarm Optimization with concept of Peer Group (DPG-PSO), which integrate the PSO with the concept of governance in human society. In DPG-PSO, population has a governor which represents the global best solution, and an opposition which takes certain properties from the governor and some are random. Instead of being influenced by only the global best solution, in DPG-PSO each particle has the choice to vote to the governor or its opposition to lead the group while in our approach all particles (even the worst) participate in choosing the global best solution not a leader to follow.

An older attempt to democratize the swarm has been introduced as Democratic PSO (DPSO) [30] which uses an extra term in the velocity update equation. The extra term represents a global best solution, in which all particles, like our method, participate in the choosing process. While in our approach, no extra term has been used but the voted global best solution replace the Pos_{gBest} solution in the equation 3, also the computing process is not the same.

4.1. Description of our proposed approach

Usually, in the PSO the global optimum solution, as described in the equation 4, is the best solution between local optima of all the particles has ever discovered[5] :

$$Pos_{gBest} = Max(Pos_{lBest}(i,T)); \forall i \in (1,N); \forall T \in (0,t) \quad (4)$$

With $Pos_{lBest}(i,T)$, as described before, is the local best solution of the i^{th} particle from the N particles, in any time T from the beginning of the optimization until the current time t. This will guide all particles to the same solution imposed by a single particle (the most efficient), which is not always necessarily the best existing solution.

SLAM is not used to be an optimization problem, because a solution which has been discovered in any time T (which represent the size of the new discovered area) would not be anymore a solution for any time $t > T$. So the PSO

used as source of inspiration, and the update role of the global best solution has been changed in order adapt it to the multi-robot exploration problem.

In our work we have democratized the swarm. The global optimum solution is calculated according to all local optimums multiplied by their objective functions as follows (equation 5):

$$Pos_{gBest} = \frac{\sum_{i=1}^{N} (OF(i) * Pos_{lBest}(i))}{N * Moy(OF)} \quad (5)$$

Wherein $Pos_{lBest(i)}$ and $OF(i)$ the local best solution (position) and the objective function value of particle i respectively, N is the number of particles within the swarm, and $Moy(OF)$ is the average value of the objective function in the whole colony.

In the collective exploration problem, the objective is to maximize the explored surface, hence, the objective function $OF(i)$ depends only on the explored area by the i^{th} robot at a time t.

The same equation of Pos_{gBest} is also as follows 6:

$$Pos_{gBest} = \frac{\sum_{i=1}^{N} (OF(i) * Pos_{lBest}(i))}{\sum_{i=1}^{N} OF(i)} \quad (6)$$

This approach will guide all particles toward a middle point (center of gravity). It will influence robots to discover the center of the map without discovering the sides (as shown in figure 2). For this reason, as describes equation 7 we changed the value of the weight of the social component of PSO w_g (which is, in most cases, a fixed value), in a way that guide robots to discover their center of gravity firstly, and, secondly the corners of the map (as shown in figure 3). To do so, $g_{BestValue}$ (which is the average value of objective functions of all robots) was compared with a threshold value $g_{bestValueThreshold}$, which is the value that decides when to switch from the first part to the second, and was divided by the maximum value that can take $g_{BestValue}$ and then, multiply it by the classic weight value that should take $w_{gStandard}$ /$w_{gClassic}$.

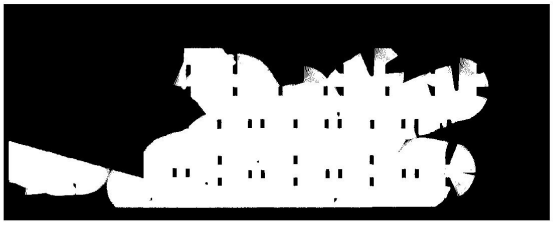

Figure 2: Explored area after 1000 iteration using Democratic RPSO with fixed weights(in the same map used in [14]).

$$w_g = \frac{g_{BestValue} - g_{bestValueThreshold}}{Maxg_{BestValue}} * w_{gClassic} \quad (7)$$

But, this will limit the value of w_g between

$$\frac{-g_{bestValueThreshold}}{Maxg_{BestValue}} * w_{gClassic}$$

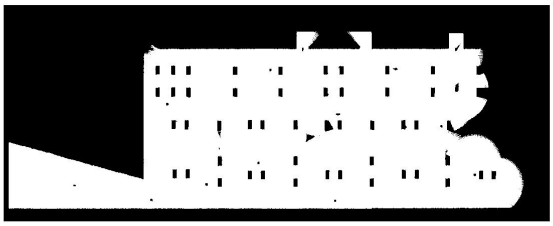

Figure 3: Explored area after 1000 iteration using Democratic RPSO with changed weights (in the same map used in [14]).

and

$$(1 - \frac{g_{bestValueThreshold}}{Maxg_{BestValue}}) * w_{gClassic}$$

In this way, the robot will be penalized by decreasing its step of advancement (which is strongly connected to the sum of weight assigned to different RPSOs variables). To adjust this limitation we, also, modified w_l (the cognitive vectors weight) as described equation 8.

$$w_l = w_{lClassic} + w_{gClassic} - abs(w_g) \quad (8)$$

with $abs(w_g)$ is the absolute value of w_g.

Thus the exploration is carried out in two steps,

1. When the value of w_g is positive, robots will tend to converge on the center of their gravity. This will helps the colony to explore their center of gravity, which is the center of the work map.
2. And when the value of w_g is negative, robots will tend to diverge far from the center of their gravity. This will helps the colony to explore the sides of the work map.

When the value of w_g is equal to zero, each robot will compute its new position without taking in consideration the social component.

4.2. DRPSO algorithm

As algorithm 1 describes, the steps of execution are listed below:

First, initialize local best position, global best position, the best position allows the robot to avoid obstacles and the best position allows the robot to maintain communication with other robots in its own position. Second, initialize all the cells of the map on the robots memory as unknown cells. After that, the robot repeats the following step in each iteration until a stopping criterion:

1. Read data from sensors and update the map in the robots memory.
2. Compute :
   - The new fitness value (*ObjectiveFunction*) of the robot. Which is the number of new discovered cells.
   - The local best position according to the concentration of the new discovered cells.

Algorithm 1: DRPSO algorithm

Initialize $Pos_{lBest}, Pos_{gBest}, Pos_{obs}$ and Pos_{Comm} as own position.
Initialize the map in the robots memory as unknown.
While stopping criteria (convergence/time) not satisfied.

1. Read sensors.
2. Update OF, Pos_{lBest} and Pos_{obs}.
3. Update Pos_{Comm}.
4. Share Pos_{lBest} and OF with other robots of the swarm.
5. Read other robots $Pos_{lBest}(i)$ and $OF(i)$ compute Pos_{gBest} and $g_{bestValue}$.
6. Compute w_g and w_l.
7. Compute $\Delta Pos(t)$.
8. Move to the new position $Pos(t+1)$.

End.

- The avoidance obstacles position, which is the symmetrical position of the obstacle relative to the robot. In other words, it is the position which allows the robot to move far away from obstacles.

3. Update the position which allows the swarm to maintain communication. Each robot should maintain communication with the nearest neighbor robot that hasnt been chosen as nearest neighbor before.
4. Share its own best position (Pos_{lBest}) and fitness value (OF) with other robots in the swarm.
5. Read other robots local best position ($PoslBest(i)$) and fitness value ($OF(i)$), and compute Pos_{gBest} according to the equation 9 and 10. With N is the number of the robots. This method calculates Pos_{gBest} with minimal iteration.

$$Pos_{gBest} = \frac{(N-1)gBest_{Value} * Pos_{gBest}}{(N-1)gBest_{Value}+1+OF(i)} + \frac{OF(i) * Pos_{lBest}(i)}{(N-1)gBest_{Value}+1+OF(i)} \quad (9)$$

$$gBest_{Value} = \frac{(N-1)gBest_{Value} + OF(i)}{N} \quad (10)$$

6. Compute w_g and w_l according to equations 7 and 8.
7. Compute Pos(t) according to equation 3.
8. move to the new position Pos(t+1) using equation 2.

5. Experimental result

In this part we will present an evaluation of the results of two simulations, in the first simulation, we used the RDPSO [14] (with Pos_{gBest} is the best solution between local optima), which is, according to [15], the most efficient algorithm. In the second simulation, we used the new method of calculating Pos_{gBest}, which we named Democratic RPSO.

In this experiment, simulations are conducted only in software simulation platform, real word constrains has not been tackled. For a real robots simulation, the problems of localization and noisy sensor inputs make a huge barrier for this purpose.

In order to ensure the loyalty of the simulation result, we simulated the two methods 20 times and calculated the average values of each method. Each simulation lasted 1480 iterations. Both simulations were run on the same map with the same initial conditions and on the same simulation platform MRSim (Multi-Robot Simulator), which is an extension of the Autonomous mobile robotics toolbox SIMROBOT developed for MatLab [2].

The constants parameters used in simulations are the following : The assigned weights to the inertia influence $w = 0.6$; The assigned weights to the local optimum solution $w_l =$ (0.42 in the RDPSO algorithm, and is calculated using equation 8 in the Democratic RPSO) ; The assigned weights to the global optimum solution $w_g =$ (0.42 in the RDPSO algorithm, and is calculated using equation 7 in the Democratic RPSO) ; The assigned weights to the avoidance obstacle component $w_{obs} = 0.18$; The assigned weights to the maintaining communication component $w_{Comm} = 0.4$; The global best threshold value $g_{bestValueThreshold} = 90$, which is the average value that could take ; The classic weight value of the global solution and the local solution are $w_{gClassic} = w_{lClassic} = 0.42$ are the same used in the RDPSO algorithm ; The maximum number of free cells that a robot can sens in our simulation experiments is $Maxg_{BestValue} = 210$. Parameters presented here are obtained empirically without giving benefit to an algorithm against other.

In these simulations, we used a colony which was formed by 10 simulation models of homogeneous robots, equipped with a laser scanner. The evaluation criterion is the area covered by the colony during a period of time T (or after number of iteration). Table 1 presents a percentage of the explored surface compared to the total area of the environment of the two methods each 40 iterations, and a comparison between the two methods.

Table 1 shows an advantage of the new method of calculating Pos_{gBest} (Democratic RPSO) all along the exploration operation. An advantage of 13.11% of the discovered area of the total environment after 360 iterations has been shown, which would translate into faster response times in a rescue operation.

Figure 4 shows the progress state (discovered area) of the two methods during the exploration time (based on the number of iteration). We can see the benefits of our new method throughout curves in figure 4, the simulation result of the RDPSO method is shown in red color, and the result of the our new method of computing Pos_{gBest}

Table 1: Discovered space with RDPSO and Democratic RPSO

Iter	RDPSO	Democratic RPSO	Dem RPSO - RPSO
40	29.06%	29.46%	0.40%
80	38.45%	39.51%	1.06%
120	46.27%	47.37%	1.09%
160	51.38%	54.15%	2.77%
200	54.36%	60.36%	6.00%
240	56.82%	65.94%	9.12%
280	59.50%	71.04%	11.54%
320	62.30%	75.08%	12.78%
360	65.29%	78.40%	13.11%
400	68.60%	81.15%	12.55%
440	71.91%	83.29%	11.38%
480	75.04%	85.01%	9.98%
520	78.16%	86.49%	8.33%
560	80.91%	87.97%	7.06%
600	83.56%	89.30%	5.74%
640	86.03%	90.48%	4.45%
680	88.04%	91.44%	3.39%
720	89.73%	92.32%	2.60%
760	91.51%	93.25%	1.74%
800	93.02%	94.10%	1.09%
840	94.22%	94.81%	0.60%
880	95.05%	95.54%	0.49%
920	95.64%	96.11%	0.46%
960	96.18%	96.59%	0.41%
1000	96.66%	97.03%	0.37%
1040	97.17%	97.38%	0.21%
1080	97.60%	97.75%	0.15%
1120	98.07%	98.00%	-0.07%
1160	98.37%	98.20%	-0.17%
1200	98.56%	98.36%	-0.20%
1240	98.70%	98.46%	-0.24%
1280	98.82%	98.52%	-0.30%
1320	98.93%	98.61%	-0.31%
1360	99.03%	98.73%	-0.30%
1400	99.10%	98.82%	-0.28%
1440	99.16%	98.89%	-0.27%
1480	99.23%	98.93%	-0.29%

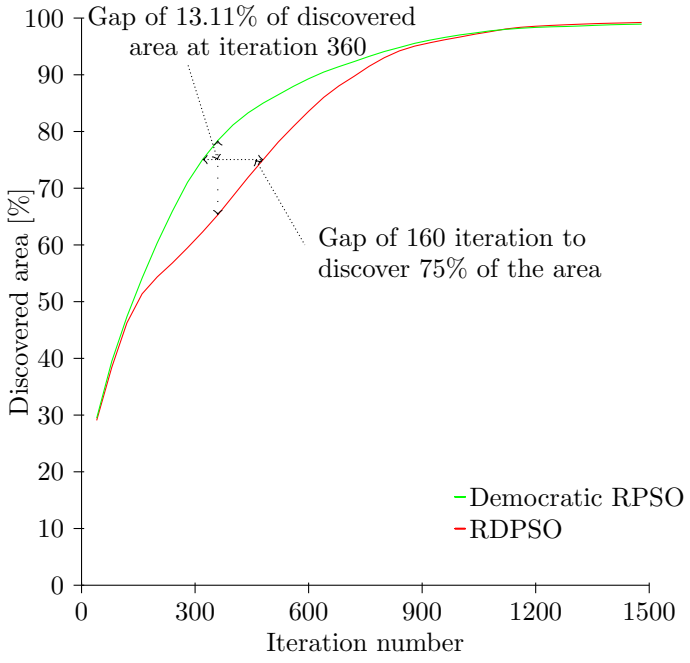

Figure 4: Discovered area using RDPSO and Democratic RPSO.

is represented in the blue curve.

As shown in Figure 4 the new method allows us to discover near 75% of the card after 320 iterations, against 480 iterations for the RDPSO method, making it a benefit of 160 iterations.

On the first steps (before 160 iterations) of the simulation there is no big difference between the two simulations, because the whole map has not been explored yet, and in each direction the robot chooses to move, it will discover a new area.

In the last iterations (after 840 iterations) both methods will end up discovering the whole map.

6. Conclusion

This paper presents a new method of calculating the global best parameter of the PSO technique to better adapt to the problem of collective exploration of an unknown environment. The idea of our method is to democratize the colony when selecting the global optimum solution.

Simulations in a virtual environment have been made, showing the benefit of our method.

Democratizing the colony saves the time (iterations) in the problems that have a variable global optimum (e.g. exploration, foraging, cleaning) but not necessarily in the problem with a fixed global optimum (e.g. solving a mathematical equation).

For future work, we will implement the democratic RPSO algorithm on physical robots in order to verify its performance in the real world constraints.

7. Acknowledgment

These research and innovation are carried out as part of a MOBIDOC thesis financed by the EU in the program PASRI.

References

[1] C. Luo, A. P. Espinosa, D. Pranantha, A. De Gloria, Multi-robot search and rescue team (2011) 296–301.

[2] M. S. Couceiro, D. Portugal, R. P. Rocha, A collective robotic architecture in search and rescue scenarios (2013) 64–69.

[3] Y. Huang, Z. Cao, S. Oh, E. Kattan, E. Hall, Automatic operation for a robot lawn mower (1987) 344–354.

[4] A. Sugiyama, T. Sugawara, Autonomous strategy determination with learning of environments in multi-agent continuous cleaning (2014) 455–462.

[5] R. C. Eberhart, J. Kennedy, A new optimizer using particle swarm theory 1 (1995) 39–43.

[6] D. F. J. K. K. Konolige, B. L. D. Schulz, B. Stewart, Distributed multi-robot exploration and mapping, Proceedings of the IEEE.

[7] W. Burgard, M. Moors, C. Stachniss, F. E. Schneider, Coordinated multi-robot exploration, Robotics, IEEE Transactions on 21 (3) (2005) 376–386.

[8] H. Choset, J. Burdick, Sensor-based exploration: The hierarchical generalized voronoi graph, The International Journal of Robotics Research 19 (2) (2000) 96–125.

[9] B. Yang, Y. Ding, Y. Jin, K. Hao, Self-organized swarm robot for target search and trapping inspired by bacterial chemotaxis, Robotics and Autonomous Systems.

[10] H. Guo, Y. Jin, Y. Meng, A morphogenetic framework for self-organized multirobot pattern formation and boundary coverage, ACM Transactions on Autonomous and Adaptive Systems (TAAS) 7 (1) (2012) 15.

[11] K. M. Wurm, C. Stachniss, W. Burgard, Coordinated multi-robot exploration using a segmentation of the environment (2008) 1160–1165.

[12] H. W. Kuhn, The hungarian method for the assignment problem, Naval research logistics quarterly 2 (1-2) (1955) 83–97.

[13] R. Arezoumand, S. Mashohor, M. H. Marhaban, Finding objects with segmentation strategy based multi robot exploration in unknown environment, Procedia-Social and Behavioral Sciences 97 (2013) 580–586.

[14] M. S. Couceiro, R. P. Rocha, N. M. Ferreira, A novel multi-robot exploration approach based on particle swarm optimization algorithms (2011) 327–332.

[15] M. S. Couceiro, P. A. Vargas, R. P. Rocha, N. M. Ferreira, Benchmark of swarm robotics distributed techniques in a search task, Robotics and Autonomous Systems 62 (2) (2014) 200–213.

[16] J. Pugh, A. Martinoli, Inspiring and modeling multi-robot search with particle swarm optimization (2007) 332–339.

[17] J. M. Hereford, M. A. Siebold, Bio-inspired search strategies for robot swarms, INTECH Open Access Publisher, 2010.

[18] K. Krishnanand, D. Ghose, Glowworm swarm optimization for simultaneous capture of multiple local optima of multimodal functions, Swarm intelligence 3 (2) (2009) 87–124.

[19] V. Gazi, K. M. Passino, Stability analysis of social foraging swarms, Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on 34 (1) (2004) 539–557.

[20] V. Braitenberg, Vehicles: Experiments in synthetic psychology, MIT press, 1986.

[21] M. Sharafi, T. Y. ELMekkawy, Multi-objective optimal design of hybrid renewable energy systems using pso-simulation based approach, Renewable Energy 68 (2014) 67–79.

[22] P. Das, H. Behera, B. Panigrahi, A hybridization of an improved particle swarm optimization and gravitational search algorithm for multi-robot path planning, Swarm and Evolutionary Computation.

[23] A. M. Pinto, A. P. Moreira, P. G. Costa, A localization method based on map-matching and particle swarm optimization, Journal of Intelligent & Robotic Systems 77 (2) (2015) 313–326.

[24] Z. Zeng, J. Xu, S. Wu, M. Shen, Antithetic method-based particle swarm optimization for a queuing network problem with fuzzy data in concrete transportation systems, Computer-Aided Civil and Infrastructure Engineering 29 (10) (2014) 771–800.

[25] S. Validi, A. Bhattacharya, P. Byrne, Integrated low-carbon distribution system for the demand side of a product distribution supply chain: a doe-guided mopso optimiser-based solution approach, International Journal of Production Research 52 (10) (2014) 3074–3096.

[26] S. Alam, G. Dobbie, S. U. Rehman, Analysis of particle swarm optimization based hierarchical data clustering approaches, Swarm and Evolutionary Computation 25 (2015) 36–51.

[27] P. K. Sahu, T. Shah, K. Manna, S. Chattopadhyay, Application mapping onto mesh-based network-on-chip using discrete particle swarm optimization, Very Large Scale Integration (VLSI) Systems, IEEE Transactions on 22 (2) (2014) 300–312.

[28] D. P. Rini, S. M. Shamsuddin, S. S. Yuhaniz, Particle swarm optimization: technique, system and challenges, International Journal of Computer Applications 14 (1) (2011) 19–26.

[29] R. Burman, S. Chakrabarti, S. Das, Democracy-inspired particle swarm optimizer with the concept of peer groups, Soft Computing (2016) 1–20.

[30] A. Kaveh, A. Zolghadr, Democratic pso for truss layout and size optimization with frequency constraints, Computers & Structures 130 (2014) 10–21.