

01 Jan 2004

# Optimal PSO for Collective Robotic Search Applications

Ganesh K. Venayagamoorthy
*Missouri University of Science and Technology*

Venu Gopal Gudise

Sheetal Doctor



## Recommended Citation

G. K. Venayagamoorthy et al., "Optimal PSO for Collective Robotic Search Applications," *Proceedings of the Congress on Evolutionary Computation, 2004. CEC2004*, Institute of Electrical and Electronics Engineers (IEEE), Jan 2004.
The definitive version is available at https://doi.org/10.1109/CEC.2004.1331059

# Optimal PSO for Collective Robotic Search Applications

Sheetal Doctor
Department of Electrical and Computer Engineering
University of Missouri - Rolla, MO 65409, USA
*skdkv6@umr.edu*

Ganesh K. Venayagamoorthy
Department of Electrical and Computer Engineering
University of Missouri - Rolla, MO 65409, USA
*gkumar@ieee.org*

Venu G. Gudise
Department of Electrical and Computer Engineering
University of Missouri - Rolla, MO 65409, USA
*venug@ieee.org*

***Abstract*- Unmanned vehicles/mobile robots are of particular interest in target tracing applications since there are many areas where a human cannot explore. Different means of control have been investigated for unmanned vehicles with various algorithms like genetic algorithms, evolutionary computations, neural networks etc. This paper presents the application of Particle Swarm Optimization (PSO) for collective robotic search. The performance of the PSO algorithm depends on various parameters called quality factors and these parameters are determined using a secondary PSO. Results are presented to show that the performance of PSO algorithm and search is improved for a single and multiple target searches.**

## I. Introduction

Mobile networks/robots are becoming increasingly popular to manage unmanned physical systems. Distributed systems are implemented in order to de-centralize the computational complexity [1]. The factors under consideration during designing include energy management, efficient communication with less disturbances, efficient computation etc. Various methods [2] explored until now have included wireless communication, image processing and vision based application, neural networks etc. Due to interference, the transmission range for the communicating bodies reduces. In order to optimize the performance, stochastic algorithms are employed at the lowermost level (robots) and only data to be sent out to the others is sent over a communication media. There are various algorithms that can be applied for these types of applications like genetic algorithms, evolutionary computational techniques etc.

A recently developed algorithm known as particle swarm optimization (PSO) that emerges and allies itself to evolutionary algorithms based on simulation of the behavior of a flock of birds or school of fish, has proven to have great potential for optimization problems. Swarm algorithms differ from evolutionary algorithms most importantly in both metaphorical explanation and how they work. What is new with the swarm algorithm is that the individuals (particles) persist over time, influencing one another's search of the problem space. The main concept is to utilize the social behavior or the communication involved in such swarms.

The strength of the PSO depends in proper selection of the parameters of the swarm, mainly the inertia weight, social and cognition components (acceleration constants). There does not exist any given set or range of parameters which is optimal for all the applications universally. This paper presents PSO based technique for determining the optimal set of parameters for a second PSO for any given application. This is demonstrated in this paper on a collective robotic search for single and multiple targets.

Section II briefly describes the particle swarm optimization. Section III describes the procedure for finding the optimal PSO parameters using a PSO. The collective robotic search application is described in section IV. Section V presents results for the single and multiple target searches with PSO.

## II. Particle Swarm Optimization

Particle swarm optimization is an evolutionary computation technique (a search method based on a natural system) developed by Kennedy and Eberhart [3] – [8]. PSO, like a generic algorithm (GA), is a population based optimization tool. However, unlike GA, PSO has no evolution operators such as crossover and mutation and moreover, PSO has less parameters. PSO is an evolutionary algorithm that does not implement survival of the fittest and unlike other evolutionary algorithms where an evolutionary operator is manipulated, the velocity is dynamically adjusted.

The system initially has a population of random solutions. Each potential solution, called a *particle*, is given a random velocity and is flown through the problem space. The particles have memory and each particle keeps track of its previous best position (*pbest*) and its corresponding fitness. There exist a number of *pbest* for the respective particles in the swarm and the particle with greatest fitness is called the *global best* (*gbest*) of the swarm. The basic concept of the PSO technique lies in accelerating each particle towards its *pbest* and *gbest* locations, with a random weighted acceleration at each time step and this is illustrated in Figure. 1, where $P(k)$ is the current position of a particle, $P(k+1)$ is its modified position, $V(k)$ is its initial velocity, $V(k+1))$ is its modified

0-7803-8515-2/04/$20.00 ©2004 IEEE

velocity, $V_{pbest}$ is the velocity considering its *pbest* location and $V_{gbest}$ is the velocity considering its *gbest* location.

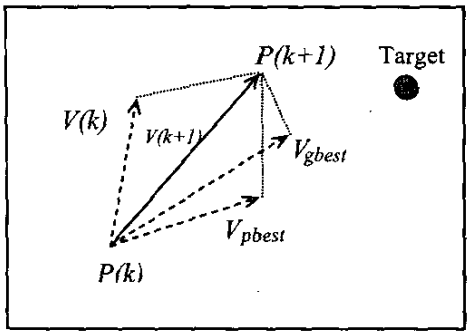


Figure 1. Concept of a swarm particle's position.

The main steps in the particle swarm optimization process are described as follows:

(i). Initialize a population of particles with random positions and velocities in *d* dimensions of the problem space and fly them.

(ii). Evaluate the fitness of each particle in the swarm.

(iii). For every iteration compare each particle's fitness with its previous best fitness (*pbest*) obtained. If the current value is better than *pbest*, then set *pbest* equal to the current value and the *pbest* location equal to the current location in the *d*-dimensional space.

(iv). Compare *pbest* of particles with each other and update the swarm global best location with the greatest fitness (*gbest*).

(v). Change the velocity and position of the particle according to equations (1) and (2) respectively. *V(k+1)* and *P(k+1)* represent the velocity and position of the $i^{th}$ particle with *d* dimensions, respectively, *rand1* and *rand2* are two uniform random functions, and W is the inertia weight, which is chosen beforehand.

$$V(k+1) = w_i * V(k) + c_1 * rand() * (P_{best}(k) - P(k)) + c_2 * rand() * (G_{best}(k) - P(k)) \quad (1)$$

$$P(k+1) = P(k) + V(k) \quad (2)$$

(vi). Repeat steps (ii) to (v) until convergence is reached based on some desired single or multiple criteria.

The parameters used in the PSO are described as follows: $w_i$ called the inertia weight controls the exploration and exploitation of the search space because it dynamically adjusts velocity. Local minima are avoided by small local neighborhoods, but faster convergence is obtained by a larger global neighborhood, and in general a global neighborhood is preferred. Synchronous updates are more costly than the asynchronous updates. *Vmax* is the maximum allowable velocity for the particles (i.e. in the case where the velocity of the particle exceeds *Vmax*, then it is limited to *Vmax*). Thus, resolution and fitness of search depends on *Vmax*. If *Vmax* is too high, then particles will move beyond a good solution, and if *Vmax* is too low, particles will be trapped in local minima. The constants $c_1$ and $c_2$ in (1) and (2), termed as cognition and social components, respectively, are the acceleration constants which changes the velocity of a particle towards *pbest* and *gbest* (generally, somewhere between *pbest* and *gbest*). The velocities of the particles determine the tension in the swarm. A swarm of particles can be used locally or globally in a search space. In the local version of the PSO, *gbest* is replaced with *lbest* and the entire process is the same.

## III. Optimal PSO

To understand the use of PSO for the communication of these particles at the ground level, the problem is broken into two parts. For simplicity of testing, first a single target case is considered and then the multiple target case. The quality factors of the PSO algorithm are the constants, inertial weight - $w_i$ and acceleration constants- $c_1$ and $c_2$. The dynamic range of $w_i$ is 0.2 to 1.2. [8] The general values for $c_1$ and $c_2$ are taken as 2. The performance of the algorithm depends on the values chosen for these parameters. Initially when the swarm starts moving, the particles are randomly oriented and therefore require a higher velocity to explore the problem space. As the area gets covered the particles start approaching the target. At this time it is essential to slow them down so that they do not overshoot the target. If $w_i$ is kept high then the new velocity will always be a large proportion of the previous one. But when the particle approaches the target it is essential that the velocity decreases and therefore by keeping a lower $w_i$ it improves the performance. Making these parameters dynamic through the entire process increases the complexity of the problem. Therefore another possible method is to find a value for each one of these constants such that the performance is optimum.

This paper looks at finding the optimum values of these parameters. The method explored here, is using the PSO algorithm to find the optimum values of these parameters. The PSO code is implemented on the program that uses PSO for target searching, where the aim is the find the optimum parameters for the target searching problem.

In this approach, a 2-level hierarchy of the swarm algorithm is used. In the lower level or the inner level, PSO is applied to get a solution to the problem at hand. In this case it is the location of the single and multiple targets. The inner swarm is therefore nothing but the actual implementation of the problem. Therefore, the PSO will function normally at this level. The only difference is that every time this loop runs, the values of the parameters $w_i$, $c_1$ & $c_2$ will change according to the outer swarm. Considering the particular application at hand (single

target and multiple target searches), the fitness function is the intensity of the source/target.

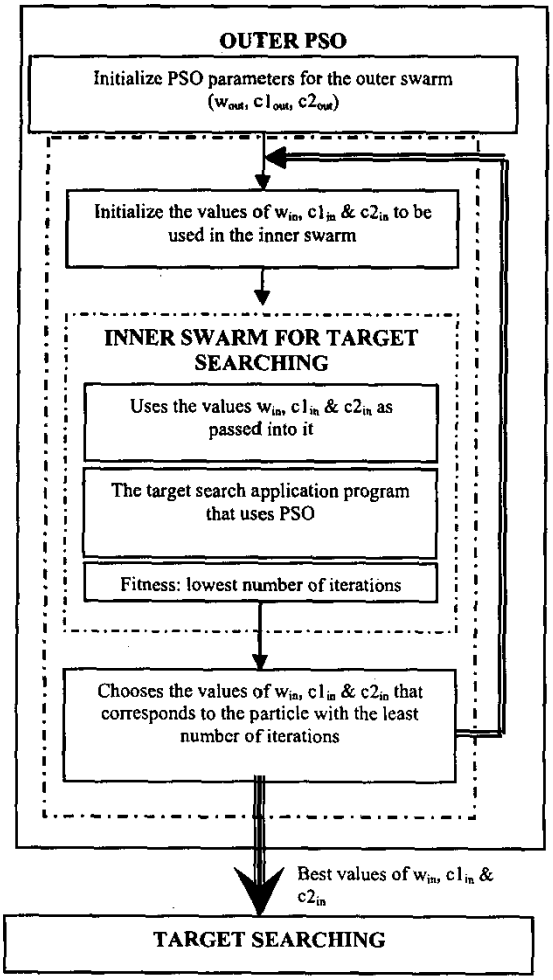


Figure 2. Flowchart for determining optimal PSO parameters ($w_{in}$, $c_{1in}$ & $c_{2in}$) using a PSO algorithm (outer PSO)

The hierarchy has been explained in figure 2. For the outer swarm, the velocity and position vectors have three values corresponding to $w_{out}$, $c_{1out}$, & $c_{2out}$, thus the solution vector which is the position vector of the gbest of the outer swarm gives the optimal set of values for the weight, cognition and social components for the application at hand. These values are again plugged back into the inner swarm and re-optimized. The process repeats until there is no change in the fitness value of the outer swarm. The fitness function for the outer swarm is the number of iterations of the inner swarm which it took to keep the error within the given limit of the inner swarm. The computation time is directly proportional to the number of the iterations taken by the inner swarm, so lesser the iterations, lesser is the computation time. The values of $w_{out}$, $c_{1out}$, and $c_{2out}$ of the outer swarm are 0.8, 2 and 2 respectively. This method is implemented on two cases to check the consistency of the program.

## IV. Target Search Problem

Collective mobile robots have become increasingly popular for target search applications [2]. One of the main reasons for this is the ease of removing human intervention. By using swarm intelligence for these applications, low cost, dispensable robots can be used to accomplish tasks that would otherwise be impossible by humans or extremely expensive. The single target and the multiple target searches are studied using the optimized PSO algorithm. The PSO particles are considered are as mobile robots and the guiding of these individual robots towards the target is carried out using the PSO algorithm. The use of the algorithm is described below for the single target and the multiple target cases.

### *A. Single Target Case*

In a given search space as shown in figure 3, the assumption is that there is a single target and the position of this target is known. The fitness function used by the PSO algorithm is the Euclidean distance of the robots from the target location, given by (3). The objective is for all the robots to converge at the target based on the individual's experience and the social interactions between the robots through the PSO algorithm.

$$fitness = \sqrt{(T_x - P_x)^2 + (T_y - P_y)^2} \qquad (3)$$

where $T_x$ and $T_y$ are the $x$ and $y$ coordinates of the target respectively, $P_x$ and $P_y$ are the $x$ and $y$ coordinates of a given individual robot respectively.

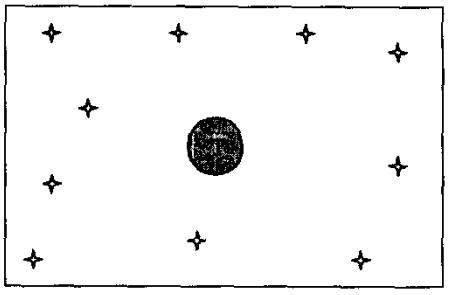

Figure 3. Graphical representation of a single target case.

For the simulation, the search space is taken as 10 units and the maximum velocity is limited to 2 units. The initial position and velocity for the robots are randomly

generated. The successive new velocities and positions are calculated using (1) and (2) respectively. Initially the robots' best position *Pbest* is the same as the initial random positions. The initial global best *Gbest* is calculated from the initial *Pbest*. This is done by calculating the Euclidean distance of the robots with the target and then searching through this array for the minimum value. The co-ordinates corresponding to this minimum value is the global best. Within a loop the algorithm calculates the new velocity depending on the parameters passed to it from the previous iteration. The new positions of the robots depend on the current velocity of the particle. After updating the position for every robot, the robots' best positions and the global best position need to be recalculated. This loop is executed until all the robots converge at the target.

### *B. Multiple Target Case*

In a given search area (figure 3), there can be a number of targets (T1, T2, T3, T4) each having a different importance. The importance of each target is based on a parameter like the intensity, the level of radiation of a radioactive source etc. These parameters define the objective of the collective robotic search and thus, fitness function to be maximized by the PSO algorithm.

When the robots start their search in the given space, they are unaware of the particular target of interest and its location in this case. Therefore, the entire swarm is divided into groups. The number of groups equals the number of targets to be explored assuming the number of targets are known. Once these groups are formed (G1, G2, G3 and G4), each group is concerned with its associated target and the aim for all the robots within the group is to converge at that target.

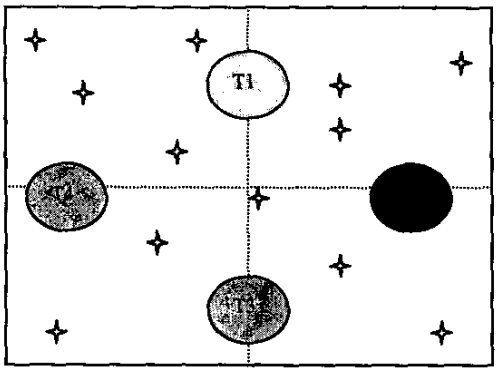


Figure 4. Graphical representation of a multiple target case.

It is taken that all the robots are equipped with four sensors to measure some intensity from the four directions as shown in figure 5, North (N), East (E), West (W) and South (S). The relative intensity calculated at each particle is given by (4).

$$Intensity = (I/d)*Cos(\theta) \quad (4)$$

Where,
$d$ = Euclidean distance between the target and the sensor on the robot, $\theta$ = the angle the target makes with the particular sensor on the robot.

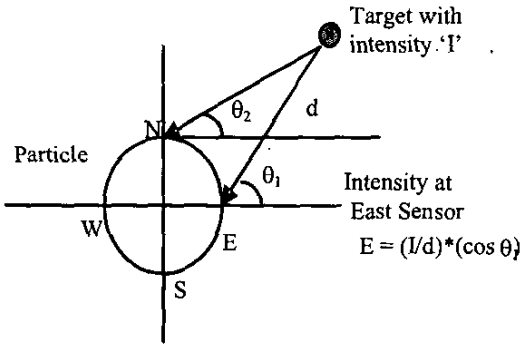


Figure 5. Shows the particle with the four sensors and the intensity readings at the sensors on each particles

The robots are then sorted into four groups corresponding to the directions. Once the sorting takes place, the operation within each group becomes independent of the robots in the other groups. Each group is concerned with reaching its target. This target is decided on the basis of the strongest directional reading on each robot. Therefore, each group acts like a local swarm by itself and they calculate a local best (lbest) within the group instead of having a global best. Once the groups have converged at their respective targets, they can read the exact location of the target (co-ordinates) and the intensity. After reading the intensity, they communicate to a common processor at a higher level which decides which of the four targets is the one of interest. It will then send the co-ordinates to the other groups and all the robots converge at that particular target.

## V. RESULTS

The optimal PSO was tested on two cases: one with single target and the other with multiple targets within the problem space. Results have been presented to show the improved performance of PSO when its own parameters have been optimized. The results have been taken as an average over 10 runs for every combination shown.

### *A. Single Target*

The commonly used values of $w_{in}$, $c_{1in}$ and $c_{2in}$ were plugged into the program and the Table below shows the performance in terms of the number of iterations required.

TABLE I. NUMBER OF ITERATIONS WITHOUT PSO OPTIMIZATION OF PARAMETER VALUES (AVERAGE OVER 10 RUNS)

| $c_{1in}$ | $c_{2in}$ | # of iterations for $w_{in}$ =0.8 |
|---|---|---|
| 2 | 2 | 491.1 |
| 0.5 | 2 | 195.4 |
| 2 | 0.5 | 186 |
| 0.5 | 0.5 | 116.4 |

By using PSO to find the optimal values of the PSO parameters, the following results were observed.

TABLE II: NUMBER OF ITERATIONS WITH PSO OPTIMIZED PARAMETER VALUES (AVERAGE OF 10 RUNS)

| Parameter Values | # of Iterations |
|---|---|
| win =0.4500<br>c1in =0.6021<br>c2in =1.500 | 115 |
| win=0.4500<br>c1in=1.1914<br>c2in=1.5284 | 123.5 |
| win=0.4500<br>c1in=0.3500<br>c2in=1.4532 | 125.2 |

As can be seen from the Table I and II above, the performance of the same code has improved by using the optimized PSO parameter values.

*B. Multiple Targets with Unknown Intensity*

The performance of the multiple target case depends on the performance of the individual groups. Therefore, it is essential to study the performance of the individual groups and the effect the parameter values have on them. Table III shows the different combinations of the parameters and average number of iterations taken over 50 runs. In this case, an inertia weight $w_{in}$ of 0.6 is used [9].

TABLE III. NUMBER OF ITERATIONS WITHOUT PSO OPTIMIZATION OF PARAMETER VALUES

| $c_{1in}$ | $c_{2in}$ | # of iterations $w_{in}$=0.6 | | | |
|---|---|---|---|---|---|
| | | G1 | G2 | G3 | G4 |
| 2 | 2 | 140 | 142 | 137 | 142 |
| 0.5 | 2 | 137 | 144 | 138 | 139 |
| 2 | 0.5 | 159 | 211 | 154 | 230 |
| 0.5 | 0.5 | 139 | 243 | 179 | 218 |

As can be seen in the Table III, there is a variation in the results between the groups especially for the last two combinations of $c_{1in}$ and $c_{2in}$. These values are chosen arbitrarily. A value set chosen may work well for one group and not for another as seen in the Table above.

One of the ways of finding optimum values for these parameters is by using PSO as described in section III. After developing a program for optimizing the parameter values, average number of iterations taken over 50 runs obtained is shown in Table IV. The average number of iterations for the local swarms to converge at their respective targets has been reduced as a result of the use of optimal PSO parameters.

TABLE IV. NUMBER OF ITERATIONS WITH PSO OPTIMIZED PARAMETER VALUES (AVERAGE OF 10 RUNS)

| Parameter values | # of iterations: Group 1 | # of iterations: Group 2 | # of iterations : Group 3 | # of iterations : Group 4 |
|---|---|---|---|---|
| $w_{in}$=0.5500<br>$c_{1in}$=1.3881<br>$c_{2in}$=2.3259 | 112 | 118 | 112 | 118 |
| $w_{in}$=0.5500<br>$c_{1in}$=0.5500<br>$c_{2in}$=2.2377 | 123 | 128 | 112 | 124 |
| $w_{in}$=0.5500<br>$c_{1in}$=2.3804<br>$c_{2in}$=2.0898 | 127 | 132 | 126 | 132 |

## VI. CONCLUSION

The PSO algorithm has proven to be quite reliable in target searching applications when the optimal values of the inertia weight and acceleration constants are determined a prior and used. The optimum values for the parameters differ for each application. Therefore, finding these values before applying the PSO algorithm to different applications ensures that the code would be executed optimally. By using this method, the performance of collective robotic search has been improved in both the single target and multiple target cases. The advantage of using a PSO to search optimal parameters is that it can be automated especially with the high speed computing capability available nowadays. This saves a lot of time guessing the right or the appropriate PSO parameters and this method requires no prior experience in PSO for the user.

The proposed method of optimization is applicable in an offline environment. The optimal parameters obtained offline thus can be used in a real time environment. In addition, it can be observed that for the two acceleration constants in PSO – $c_1$ and $c_2$, the optimal PSO finds a higher value for $c_2$ than for $c_1$. This observation emphasizes that the social component/social interaction plays the major role in PSO. Future work involves rigorous experimentation on a number of applications and evolving the number of swarms/groups for a particular landscape as optimally required.

## REFERENCES


[1] Z. Butler, D. Rus, "Event-based motion control for mobile-sensor networks", *IEEE Pervasive Computing,* vol. 2, Issue: 4 , Dec. 2003, pp. 34 – 42.

[2] M. Haenggi, "Mobile sensor-actuator networks: opportunities and challenges", *Proceedings of the 7th IEEE International Workshop on Cellular Neural Networks and Their Applications*, July 22-24, 2002, pp. 283 – 290.

[3] J. Kennedy and R. Eberhart, "Particle swarm optimization," *Proceedings, IEEE International Conference on Neural Networks*, vol. 4, pp. 1942–1948.

[4] J. Kennedy, R Eberhart and Y. Shi, *Swarm intelligence*, Morgan Kaufmann Publishers, 2001.

[5] R. Eberhart and Y Shi, "Particle swarm optimization: developments, applications and resources," *Proceedings of the 2001 Congress on Evolutionary Computation*, vol. 1, 2001, pp. 81 -86.

[6] Y. Shi and R. Eberhart, "Empirical study of particle swarm optimization," *Proceedings of the 1999 IEEE Congress on Evolutionary Computation*, vol. 3, 1999, pp. 1945 - 1950.

[7] Y. Shi and R. Eberhart, "A modified particle swarm optimizer," *IEEE International Conference on Evolutionary Computation*, 1998, pp. 69 –73.

[8] Y. Shi and R. Eberhart, "Parameter selection in particle swarm optimization," *Proceedings of the Seventh Annual Conference on Evolutionary Programming*, 1998, pp. 591-601.

[9] S. Doctor, G. K. Venayagamoorthy, "Unmanned vehicle navigation using swarm intelligence," *Proceedings of the International Conference on Intelligent Sensing and Information Processing*, Chennai, India, January 4-7, 2004, pp. 249 – 253.