# Improving the Performance of Particle Swarm Optimization Algorithm With a Dynamic Search Space

Benoît Vallade, Tomoharu Nakashima
Department of Computer Science and Intelligent Systems Osaka Prefecture University
Osaka, Japan
valladeben@cs.osakafu-u.ac.jp & tomoharu.nakashima@kis.osakafu-u.ac.jp

***Abstract*-This paper addresses an improvement idea for Particle Swarm Optimization Algorithm (PSO). As a search algorithm, the PSO is used to tune a set of parameters and find the best combination of parameter values for this set. These parameters habitually take their values in a static search space. This paper proposes a solution to improve the efficiency of the algorithm with optimization problems using parameters, which take their values in dynamic space. The appreciable experiments' results prove that this one is an efficient solution to such problems.**

***Keywords-algorithm of non-deterministic search; particle swarm optimization algorithm; dynamic search space***

## I. INTRODUCTION

Nowadays, in the aim to solve optimization problems, the algorithms of non-deterministic search are commonly used. These problems, such as robots' motion optimization [1], require to find the best combination of parameter values for the particular problem at hand. There are various kinds of search algorithms [2], such as tabu algorithms, genetic algorithms, PSO (Particle Swarm Optimization) algorithms, and others. Among all these algorithms, this paper will focus on the particle swarm optimization algorithm also called the PSO algorithm. This choice has been motivated by the high degree of adaptability of this algorithm which is the best choice to implement our improvement.

The concept of the PSO algorithm is based on the simulation of a simplified social model and more particularly on the animals flocking [3]. Its conception follows some standard which have evolved overtime [4].

Like the other algorithms of non-deterministic search, these standard PSO algorithms allow to tune a set of parameters, which take their values in static search space.

This means that any time during the optimization, the search space of each variable will stay the same.

However, some optimization problems use a set of variables, which take their values in dynamic search spaces [1]. This means that the search spaces of the variables may vary during the optimization.

This paper presents our solution to improve the efficiency of the PSO algorithms in case of problems using variables with dynamic search space. First, in the next section, we will describe the global concept of the search algorithms and detail the standard versions of the PSO algorithm. Next, the third section will explain in details the particularities of these dynamic problems and the algorithm's improvement used to solve them. The fourth section will give the results of some experiments which compare the efficiency of both algorithms, standard and new, on these problems. Finally, we will conclude on the quality of the PSO and the efficiency of the new algorithm.

## II. STANDARD PARTICLE SWARM OPTIMIZATION

### *A. An algorithm of non deterministic search*

As introduced before, the PSO algorithm is an algorithm of non-deterministic search. This means that it searches for the best combination of values for a set of variables. As the Table I shows, the variables take their values in search spaces defined by a minimal and a maximal values. These limits are given by the user and will take constant values.

TABLE I. SET OF VARIABLES' STATIC SEARCH SPACES

| | Min | Max |
|---|---|---|
| A | -5 | 5 |
| B | 2 | 6 |
| C | -10 | -5 |
| D | 0 | 10 |
| E | -2 | 5 |

In addition, it means that the algorithm follows the same global processes. Firstly, the algorithm generates a set of random solutions. A solution is a combination of values for the set of parameters. After that, the solution's quality will be determined through a fitness function. This function is completely dependent on the problem to be optimized and is given by the user. This quality value is used to compare the actual solution to the precedent best solution, and a new solution will be generated. These three steps (calculate solution's quality, compare solutions and generate a new solution) will be repeated so long as the optimization continues. This one stops when the stop criterion satisfies certain criteria chosen by the user (time, number of iterations, etc.). To finish, the generation of the new

solution depends on the algorithm (tabu search, genetic algorithm, PSO), but it generally uses the best and previous solutions.
This process is described in the Figure 1.

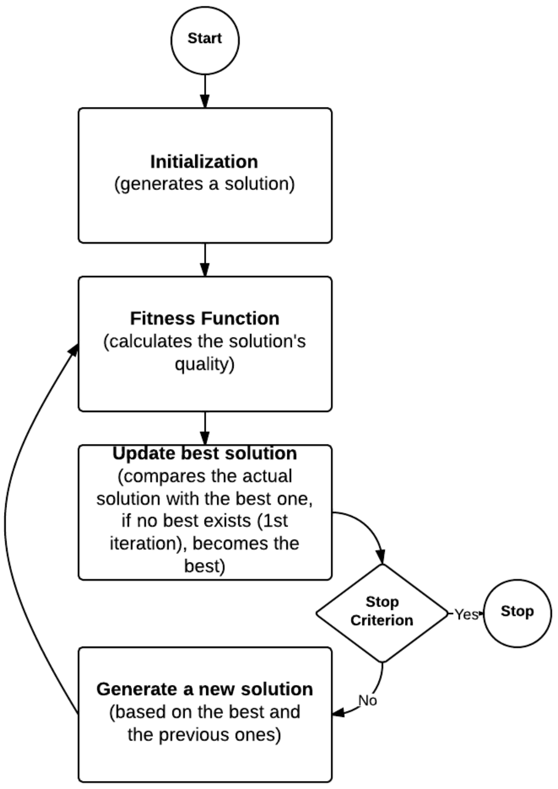
Start
Initialization
(generates a solution)
Fitness Function
(calculates the solution's quality)
Update best solution
(compares the actual solution with the best one, if no best exists (1st iteration), becomes the best)
Stop Criterion
Yes
Stop
No
Generate a new solution
(based on the best and the previous ones)

Figure 1. Global algorithm of an algorithm of non-deterministic search.

## B. *Concept overview*

The special feature of the PSO algorithm is that its concept is based on the animals flocking [3].

As explained above, a solution is a combination of values for a set of parameters, while a problem is defined by these parameters, which take their values in static search spaces. Consequently, another way to represent an optimization problem is to consider a solution as a position in a search space defined by crossing all the individual parameters' search space. The position takes place in a hyper-space of dimension equals to the numbers of parameters. And its coordinate's values correspond to the parameter's value of the solution.

Coming back to the animal's social behaviour subject and more particularly on the birds flocking; during their feeding time, multiple birds evolved in the same space and search for the position where there is the biggest quantity of foods. In the course of their search, each bird always remembers the position where they have found the biggest quantity of food. In addition, as the birds follow a social behaviour inside the flock, they also share the best position found by the whole flock. Finally, as shown in Figure 2, each bird adapts its movements in the search space according to these knowledge.

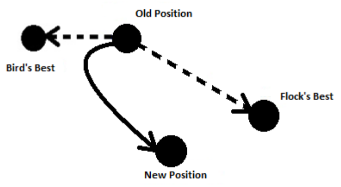

Figure 2. Birds' movements according to Best Positions knowledge.

This social behaviour is used by the PSO algorithm as a conceptual idea to generate new solutions and optimize the set of parameters. In this transposition, the birds will be called particles, the flock will be the swarm and the quantity of foods will correspond to the quality of the solution. As the flock of birds seeks for the best food's position, the swarm of particles seeks for the best quality's position.

## C. *Standard algorithm*

Our research is based on 2011's version of the standard PSO algorithm described in the paper of Maurice Clerc [4], with the particularity of not using the neighbourhood system (in case of neighbourhood system, the particles are grouped in teams and they share the information about the best position found by all the team's member only inside the team, in our case there is only one big team, which correspond to the whole swarm). This part of the paper gives some details about this version of standard PSO algorithm.

*1) Particle's components and algorithm:* As explained in the previous part on the birds flocking transposition, a swarm of particles is included in the search space. Each particle is aware of:

- Its Position (initialized randomly in the search space)
- Its Velocity (initialized randomly in the search space)
- Its Best Position ever found (initialized as the first particle's position)
- The Swarm's Best Position (initialized by comparing all the quality Particle's first position)

It should be noted that in the 2011 version, the initialization of the positions' and velocities' values are randomly generated, parameter by parameter.

Each iteration of the optimization, these particles' attributes are updated following the process described in Figure 3:

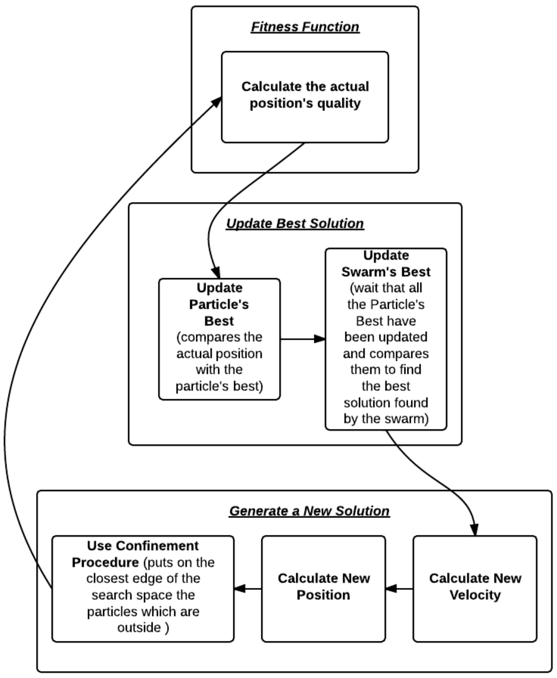
Fitness Function
Calculate the actual position's quality
Update Best Solution
Update Particle's Best (compares the actual position with the particle's best)
Update Swarm's Best (wait that all the Particle's Best have been updated and compares them to find the best solution found by the swarm)
Generate a New Solution
Use Confinement Procedure (puts on the closest edge of the search space the particles which are outside )
Calculate New Position
Calculate New Velocity

Figure 3. Process of an iteration of the PSO algorithm.

2) *Evolution rules:* Below are given the equations used for the velocity and position update and all the other details useful to the implementation of the 2011 version of the PSO algorithm.

a) *Swarm size and initialization:* In the 2011 version the swarm size (number of particles) will be user defined, with 40 as suggested value. The initialization of the particles' attributes will be conducted as described before.

b) *Calculate Velocity and Position:* First the velocity will be updated by using the following equations:

$$G_i = x_i + c * \left(\frac{p_i + l_i - 2x_i}{3}\right) \tag{1}$$

$$H_i(G_i, \|G_i - x_i\|) \tag{2}$$

$$v_i(t+1) = wv_i(t) + x'_i(t) - x_i(t) \tag{3}$$

In these equations:

- c = 1.193 and w=0.721 (constants)
- $x_i$ (or $x_i(t)$) is the actual position value of the $i^{th}$particle
- $p_i$ is the particle's best position of the $i^{th}$particle
- $l_i$ is the swarm's best position of the $i^{th}$particle
- $G_i$ is the centre of gravity of the three points $x_i$, $p_i$ and $l_i$.
- $H_i$ is the hyper sphere of centre $G_i$ and radius $G_i - x_i$
- $x'_i(t)$ is a position randomly choose in the hyper sphere $H_i$
- $v_i(t)$ is the actual velocity of the $i^{th}$particle
- $v_i(t+1)$ is the new velocity of the $i^{th}$particle

Next, the position is updated by using the equation:

$$x_i(t+1) = wv_i(t) + x'_i(t) \tag{4}$$

In cases where $l_i = p_i$, the following gravity centre equation is used for the velocity updating:

$$G_i = x_i + c * \left(\frac{p_i - x_i}{2}\right) \tag{5}$$

c) *Confinement:* But, sometimes, the new position of the particle is out of the search space. In those cases, the algorithm uses a confinement procedure, which moves the particle on the closest edge of the search space. This movement is conducted by replacing the value of each parameter of the position by the closest corresponding parameter's search space limits, min or max. Finally, the velocity forced to the following value:

$$v_i(t+1) = -0.5 * v_i(t+1) \tag{6}$$

d) *Particle's and Swarm's Best:* To finish the quality of the new position is calculated by using the fitness function. As said before, this function depends on the problem and is defined by the user. Its results will be used to compare the different positions found by the algorithm. During a first comparison the value of the particle's best position is updated in function of the previous one and of the actual position. In order to do the second comparison, the algorithm waits that all the particles' best of the swarm are updated. All the particles' best position of the swarm will be compared to determine which the swarm's best position is, and this knowledge will be shared with all the swarm's particles.

## III. PSO IN DYNAMIC SEARCH SPACES

This section discusses the topic of the problems based on set of parameters with dynamic search spaces. And then the proposed solutions to deal with such problems and through the changes made on the standard PSO algorithm are explained.

### A. *Dynamic Search Space problems*

Contrarily to the previous standard types of problems, which used parameters taking their values in static search spaces, some problems are based on dynamic search spaces. In these kinds of problems, the search spaces limit of some parameters depends on the value of other parameters. Table II gives an example of such a set of parameters:

TABLE II. SET OF VARIABLES' DYNAMIC SEARCH SPACES

| | Min | Max |
|---|---|---|
| A | 0 | 5 |
| B | -A | A |
| C | -5*A+B | +5*A+B |
| D | -15 | 2*A |
| E | -20 | 10 |

In those kinds of problems, commonly used in robots' motion optimization [1], the search spaces limits depends on the value of the parameters and consequently on the position of the particle. Finally, as shown in Figure 4, all the particles evolved in a different search space, which changes in function of the particle's position. However, a maximal space search can be created by using the maximum value possible for each parameter's maximum limit and the minimum value possible for the parameter's minimum limit.

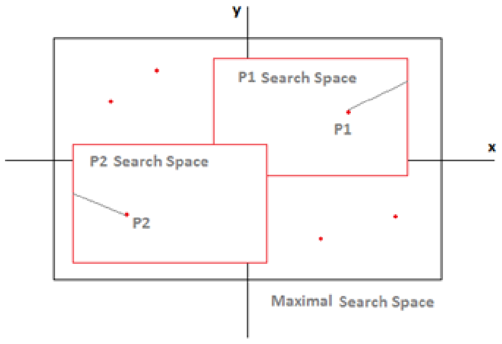

Figure 4. Dynamic search space representation in a two-parameter optimization problem.

Table III uses the Table II 's example to create the corresponding maximal search space.

TABLE III. SET OF VARIABLES' MAXIMAL SEARCH SPACES

| | Min | Max |
|---|---|---|
| A | 0 | 5 |
| B | -5 | 5 |
| C | -30 | 30 |
| D | -15 | 10 |
| E | -20 | 10 |

This space consequently contains all the individual search spaces possible. By individual search space, we mean the search spaces which are created by using the position value. But it also contains positions which are not included in these individual search spaces. To keep the precedent example (Table II and Table III), the position of coordinate (0;-5;0;0;2) is available in the maximal search space but not in individual spaces. As the search spaces limits are user-defined, we will call these positions, "uninteresting positions".

## B. *Dynamic search space PSO concept*

To solve such problems, there are two options. The first one is to apply the optimization on the maximal search space. As described above, this space includes all the search spaces possible and has the particularity to be static. The advantage of such a solution is that, as the search space is static, the standard PSO algorithm described before can be used. The disadvantage is that the optimization will also be conducted on uninteresting positions. This may result in loss of time and a final optimization position not intended by the user.

The second solution is to use individual search spaces. This means that each particle will have its own search space and this one will change as the same time as the particle move. This solution avoids the search on the uninteresting positions but implies some modifications to the standard algorithm.

As we chose to improve the efficiency of the PSO algorithm in case of dynamic search space problems, we will explain in the following part the necessary changes to the standard algorithm.

## C. *Dynamic search space PSO modifications*

This part discusses about the problem faced by the standard algorithm resulting from the choice of the second solution and about the possible modifications to avoid it.

1) *Problem:* During the initialization step as well than during the confinement methods, the new value of the position will be generated parameter by parameter. A random value will be generated between the parameter's search space limits for the initialization and the closest limit will be searched for the confinement. But, by using dynamic search spaces, these limits values will depend on others parameters' values. Also, the algorithm would not be able to generate a parameter's value if its limits have not ever been defined. That is why the parameters' values need to be defined in the good order.

2) *Modification:* This order will of course be based on the links between the parameters. The parameter A is linked with the parameter B if value of B is required to calculate the limits of A's search space.

In the aim to represent these links, we chose to use an acyclic graph representation.

These graphs are tree graphs with the particularity to allow multiple roots and multiple parents for a same child. Of course we can't allow cycle due to the impossibility to generate a position value if the parameters are linked through a cycle. In this case, the first proposed methods using the maximum search space should be used.

Our implementation uses a unique root, which does not correspond to any parameter, but it allows us to insert all the parameters in the same graph, even the one which are not linked with others parameters (search space limits have constant value). These special parameters will be direct leaf of the root.

Each parameter is represented by a node which is linked to other nodes following the parameters links. The node A is parent of the node B if the parameter A needs the value of B. The nodes will be sorted in depth layers, a node will also be in the layer under the layer of its deepest parents. The direct leafs are inserted in the deepest layer of the graph.

Finally, to generate the position value, the order to follow is decided by using the graph. This path corresponds to a back breadth-first search of the graph. This means that we start from leafs and we head to the root of the graph by visiting each node of a layer before to go to the upper one. node includes calculate its search space limits and generate its value (or search the closest limits for the confinement method). Figure 5 is the acyclic graph representation of the example given in Table II. It also shows the calculation path (dotted arrows):

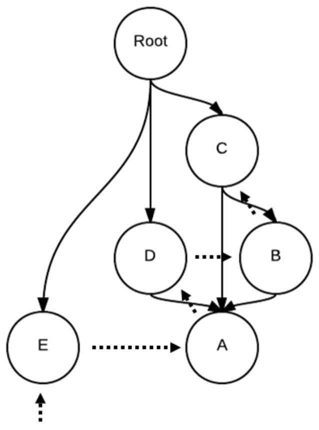

Figure 5. Acyclic graph representation of parameters link and calculation path.

## IV. EXPERIMENTS

In order to compare the efficiency between the standard and the new algorithm on dynamic search spaces problems, we set up two experiments. The problem is that there are no such problems in literature [5]. Consequently, we cannot compare it with the other algorithms' results but have to create our own artificial problems. This is the why the two next experiments do not correspond to any known problems and have no real correspondence with the real life. These problems have only been designed to proof the functionality and efficiency of the new algorithm compared with the standard algorithm.

### A. *Experiment 1*

For this one, we will use a swarm of 40 particles and launch optimization of 500 iterations. We created three optimization problems very simple (so they will be solved in 500 iterations) and compared how many iterations are required to find the optimal position (position with the best fitness quality).

These problems have the following characteristics:

- two dimensional problems (parameters: X , Y)
- The Y's search space limits depends on X's value
- The optimal position is the point of coordinate (5.5 ; 0.01) and the fitness function calculate the distance between the particle position and the optimal position.
- X's value vary in [0 ; 1000]
- Y's vary in the limits defined by Table IV, the standard algorithm will use the maximum search space:

TABLE IV. SEARCH SPACE LIMITS FOR THE Y PARAMETER

| | New Algorithm | | Standard Algorithm | |
|---|---|---|---|---|
| | Min | Max | Min | Max |
| A | $-(0.05 * x) + 0.25$ | $+(0.05 * x) - 0.25$ | -49.75 | 49.75 |
| B | $-(0.25 * x) + 1.25$ | $+(0.25 * x) - 1.25$ | -240.5 | 240.5 |
| C | $-(0.5 * x) + 2.5$ | $+(0.5 * x) - 2.5$ | -497.5 | 497.5 |

We repeated the optimization 10 times and took the average number of iterations needed to find the optimal point (as we calculate the distance, the fitness value = 0). Table V regroups the results of this experiment:

TABLE V. AVERAGE NUMBER OF ITERATIONS NEEDED TO FIND THE OPTIMAL POSITION

| | A | B | C |
|---|---|---|---|
| Standard algorithm | 279 | 297 | 298 |
| New algorithm | 282 | 255 | 250 |

We remark that the standard algorithm has better results for the problem A, but becomes less efficient on B and C. This means that the new algorithm would be more efficient on big search spaces, and more particularly, when the number of uninteresting positions grows up, which make sense.

### B. *Experiment 2*

As the previous results seems indicate that the new algorithm is more efficient on big search space we set up a second experiment to confirm. To do so we still used the same configuration for the PSO algorithm (40 particles, 500 iterations and we created a more complicated problem evolving on a bigger maximal search space. The search spaces limits of this problem are described in Table VI and the parameters links can be visualized in Figure 6.

TABLE VI. SEARCH SPACE LIMITS FOR THE SET OF PARAMETERS

| Parameters | Min | Max |
|---|---|---|
| A | -(D+E+F) | D+E+F |
| B | -(E+F+G) | E+F+G |
| C | -(H+I) | H+I |
| D | -500 | 500 |
| E | -2.5*J | 2.5*J |
| F | -0.5*J | 0.5*J |
| G / H | -500 | 500 |
| I | -(K*0.5) | K*0.5 |
| J / K / L | -500 | 500 |

It should be noted that the maximal search space is not given, but it can be easily calculated, as shown in the previous examples.

Also to be noted that the optimal position is a static position (0,0...,0), centre of the search space. The fitness function calculates the distance to this point, so the best quality value possible is 0.

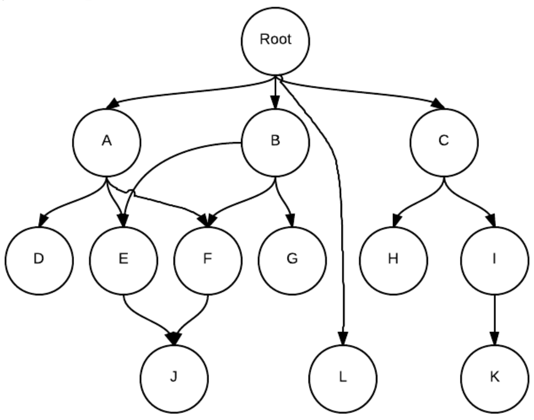

Figure 6. Acyclic graph representation of parameters link.

In this experiment, the problem is too big to be solved in 500 iterations; so, we compare the quality of the solutions. The standard algorithm gave an average of 0.0027, while the new algorithm gave an average of $7.90 * 10^{-25}$. The results of this experiment show clearly the efficiency of the new algorithm.

## V. CONCLUSION AND FUTURE WORK

To conclude, the PSO is a very simple and easily adaptable algorithm. The actual standard version of the PSO algorithm is able to deal with dynamics search space by venturing a loss of time and falling on an uninteresting result. This paper described an efficient solution to improve its performance in this case. However, there is actually no efficient solution for dynamic search space problems, where parameters are cycled linked. Our future objectives will be to be able to deal with cycled graph, and to test our solution on real world problems.

## REFERENCES

[1] T. Uchitane and T. Hatanaka, “Applying evolution strategies for biped locomotion learning in roboCup 3D soccer simulation”, Proc. of 2011 IEEE Congress on Evolutionary Computation New Orleans. LA, 2011, pp. 179-185.
[2] H. Youssef S. M. Sait, and H. Adiche, “Evolutionary algorithms, simulated annealing and tabu search: a comparative study”, Engineering Applications or Artificial Intelligence, 2001, pp. 167-181.
[3] J. Kennedy and R. Eberhart, “Particle swarm optimization”, Proc. of IEEE International Conference on Neural Networks Perth, 1995, pp. 1942-1948.
[4] M. Clerc, “Standard particle swarm optimisation”, technical report, 2012.
[5] M. Molga and C. Smutnicki, “Test functions for optimization needs”, 2005.