

## Solving Non Linear Function with Two Variables by Using Particle Swarm Optimization Algorithm

Ahmed Shawki Jaber 

Received on: 27/10/2010

Accepted on: 5/1/2011

### Abstract

The meaning of the Particle Swarm Optimization (PSO) refers to a relatively new family of algorithms that may be used to find optimal (or near optimal) solutions to numerical and qualitative problems.

The genetic algorithm (GA) is an adaptive search method that has the ability for a smart search to find the best solution and to reduce the number of trials and time required for obtaining the optimal solution.

The aim of this paper is to use the PSO to solve some kinds of two variables function which submits to optimize function filed. We investigate a comparison study between PSO and GA to this kind of problems. The experimental results reported will shed more light into which algorithm is best in solving optimization problems.

The work shows the iteration results obtained with implementation in Delphi version 6.0 visual programming language exploiting the object oriented tools of this language.

### حل معادلة غير خطية ذات متغيرين باستخدام خوارزمية أمثلية السرب الجزيئي

#### الخلاصة

أن تحقيق أمثلية السرب الجزيئي Particle Swarm Optimization (PSO) تعني الإشارة الى عائلة جديدة من الخوارزميات التي تستخدم لاجاد حلول مثالية (أو أقرب الى المثالية) للمسائل العددية والكمية.

الخوارزمية الجينية Genetic Algorithm (GA) هي طريقة متكيفة لها القابلية على إجراء بحث كفاء لإيجاد الحل الأفضل وتقليل عدد المحاولات والزمن المطلوبين للحصول على الحل الأمثل.

البحث يهدف إلى استخدام أمثلية السرب الجزيئي (PSO) لحل بعض انواع الدوال التي تعتمد متغيرين الخاضعة الى حقل أمثلية الدوال.

وقد تم تحقيق دراسة مقارنة بين طريقة (PSO) والخوارزمية الجينية لحل هذا النوع من المسائل. ان النتائج العملية المستخلصة من البحث تسلط الضوء على أي الخوارزميتين افضل في حل مسائل أمثلية الدوال. لقد نفذ البرنامج الخاص بالدراسة بلغة دلفي (Delphi) من الجيل السادس، وهي من لغات البرمجة المرئية.

### 1. Introduction

Genetic algorithms (GAs) are a part of evolutionary computing, which is a rapidly growing area of artificial intelligence. GAs are inspired by Darwin's theory

about evolution. Simply said, solution to a problem solved by GAs is evolved.

GAs were first suggested by John Holland and developed by him and his students and colleagues in the

seventies of last century. This led to Holland's book "Adaptation in Natural and Artificial Systems" published in 1975 [1]. Over the last twenty years of the last century, it has been used to solve a wide range of search, optimization and machine learning problems. Thus, the GA is an iteration procedure, which maintains a constant size population of candidate solutions [1]. In 1992 John Koza has used GA to evolve programs to perform certain tasks. He called his method "genetic programming" (GP) [2].

One of the important new learning methods is the Particle Swarm Optimization (PSO), which is simple in concept, has few parameters to adjust and is easy to implement. PSO has found applications in a lot of areas. In general, all the application areas that the other evolutionary techniques are good at are good application areas for PSO [3].

In 1995, Kennedy J. and Eberhart R. [4], introduced a concept for the optimization of nonlinear functions using particle swarm methodology. The evolution of several paradigms outlined, and an implementation of one of the paradigms had been discussed

In 1999, Eberhart R.C. and Hu X. [5], arranged a new method for the analysis of human tremor using PSO which is used to evolve a Neural Network (NN) that distinguishes between normal subject and those with tremor.

In 2004, Shi Y. [3], surveyed the research and development of PSO in five categories: algorithms, topology, parameters, hybrid PSO algorithms and applications. There are certainly other research works on PSO which are not included here due to the space limitation.

## 2. Genetic Algorithm

Genetic Algorithms (GAs) are search algorithms based on the mechanics of natural selection and natural genetics. They combine survival of the fittest among string structures with a structured yet randomized information exchange to form a search algorithm with some of the innovative flair of human search. In every generation, a new set of artificial creatures (strings) is created using bits and pieces of the fittest of the old; an occasional new part is tried for good measure. While randomized GAs are no simple random walk, they efficiently exploit historical information to speculate on new search points with expected improved performance [6].

## 3. Optimization of a Function Problem [7]

There is a large class of interesting problems for which no reasonably fast algorithms have been developed. Given such a hard optimization problem it is often possible to find an efficient algorithm whose solution is approximately optimal. We discuss the basic features of a GA for optimization of a simple function. Let  $f(x_1, x_2) = 21.5 + x_1 \cdot \sin(4\pi x_1) + x_2 \cdot \sin(20\pi x_2)$  (1)

where  $-3.0 \leq x_1 \leq 12.1$  and  $4.1 \leq x_2 \leq 5.8$ .

Since  $x_1, x_2$  are real numbers, this implies that the search space can be huge and that traditional methods can fail to find optimal solution [7].

## 4. Implementation of GA in Optimization of a Function Problem

### 4.1. Problem Representation [8]

To apply the GA for maximizing  $f(x_1, x_2)$  in (1), a genetic representation of solution to the problem must be appropriately chosen

first. The Simple GA uses the binary representation in which each point  $(x_1, x_2)$  is described by a chromosome vector coded as a binary string. We use a binary vector as a chromosome to represent real values of the variable  $x$ , the length of the vector depends on the required precision, which in this example, is six places after the decimal point.

The domain of the variable  $x_1$  has length 15.1; the precision requirement implies that the range  $[-3.0, 12.1]$  should be divided into at least  $15.1 \times 10000$  equal size ranges. This means that 18 bits are required as the first part of the chromosome:

$$217 < 151000 < 218$$

The domain of the variable  $x_2$  has length 1.7; the precision requirement implies that the range  $[4.1, 5.8]$  should be divided into at least  $1.7 \times 10000$  equal size ranges. This means that 15 bits are required as the second part of the chromosome:

$$214 < 17000 < 215$$

The total length of a chromosome (solution vector) is then  $m = 18 + 15 = 33$  bits;

the first 18 bits code  $x_1$  and the remaining 15 bits from (19–33) code  $x_2$  [8].

Let us consider an example chromosome:

(01000100101101000011111001010010) corresponds to  $(x_1, x_2) = (-2.334465, 4.699438)$ .

The fitness value for this chromosome is:

$$f(-2.334465, 4.699438) = 26.566770.$$

#### 4.2. Initial Population and Evaluation Function [6]

To optimize the function  $f$  using GA, we create a population of  $pop\_size$  chromosomes. All 33 bits in all chromosomes are initialized randomly.

Evaluation function for binary vector  $v$  is equivalent to, the function  $f$ :

$$eval(v) = f(x_1, x_2) \quad (2)$$

where the chromosome  $v$  represents the 33 digits string.

During the evaluation phase we decode each chromosome and calculate the fitness function from  $(x_1, x_2)$  values just decoded.

For example, the two chromosomes:

$$v_1 =$$

$$(10011010000000111111101001101111),$$

$$v_2 =$$

$$(111000100100110111001010100011010),$$

Correspond to values  $x_1$  and  $x_2$  respectively. Consequently, the evaluation function would rate them as follows:

$$eval(v_1) = f(11.161431, 4.954643) =$$

$$34.237697$$

$$eval(v_2) = f(10.953948, 7.767766) =$$

$$33.832967$$

Clearly, the chromosome  $v_1$  is the best of the two chromosomes, since its evaluation returns the highest value.

#### 4.3. Genetic Operators [6]

##### 1. Selection Operator

Roulette wheel is chosen to sum up the fitness' of all individuals and to calculate each individual percentage of the total fitness. The percentage of the total fitness of each individual is then used as the probability to select  $N$  individuals from the set population and copy them into the set selected-parents

##### 2-The Mating Crossover Operator

Individuals from the set selected-parents are mated to generate offspring's for the next generation. The two parents generate two offspring's using a crossover operation. For this example, to illustrate the crossover operator on chromosome with a crossover with probability  $P_c$ , we generate random integer number ( $pos$ ) from the range

1...32. The number (pos) indicates the position of the crossing point. Suppose the pair of chromosomes is:  
 $v1 =$   
 (100110100000001111111010011011111),  
 $v2 =$   
 (000010000110010000001010111011101),  
 and the generated number (pos)=9. These chromosomes are cut after the 9th bit and the remaining 24 bits exchange position:  
 the two resulting offspring's are:

$v1' =$   
 (100110100011001000001010111011101),  
 $v2' =$   
 (000010000000001111111010011011111),

### Mutation Operator

Mutation is a random change of one or more genes (positions in a chromosome) with a probability equal to the mutation rate  $P_m$  a gene is changed/swapped, i.e  $0 \rightarrow 1$  and  $1 \rightarrow 0$ . The probability for a mutation is usually kept small. Assume that the fifth gene from the  $v2$  chromosome was selected for a mutation. Since the fifth gene in this chromosome is 1, it would be flipped into 0. So the chromosome  $v2$  after this mutation would be:

$v2' =$   
 (00000000000000111111101001101111).

## 2. Genetic Parameters

For this particular problem, Michalewicz [8] used the following parameters: population size  $pop\_size=20$ , probability of crossover  $c=0.25$ , probability of mutation  $P_m=0.01$ .

### 4.1 Experimental Results

In Table (1) below we provide the generation number for which we noticed the improvement in the evaluation function, together with the value of the function.

For this problem, a simulation has been

constructed in order to apply the GA, using population size  $pop\_size=20$ , the crossover parameters mentioned above, the

following results are being obtained:

$v_{max} =$   
 (01001111111111111010011111111111)  
 Which corresponds to a value  $(x1, x2) = (12.099251, 5.799325)$ , and  $f(v_{max}) = 35.761033$ .

Notice that the solution  $f(v_{max}) = 35.761033$  is obtained in the generation (660)

## 5. Particle Swarm Optimization (PSO)

PSO was originally developed by a social-psychologist J. Kennedy and an electrical engineer R. Eberhart in 1995 and emerged from earlier experiments with algorithms that modeled the "flocking behavior" seen in many species of birds. Where birds are attracted to a roosting area in simulations they would begin by flying around with no particular destination and then spontaneously forming flocks until one of the birds flew over the roosting area [9]. PSO has been an increasingly hot topic in the area of computational intelligence. it is yet another optimization algorithm that falls under the soft computing umbrella that over genetic and evolutionary computing algorithms as well [10].

### 5.1. Fitness Criterion

One of these stopping criteria is the fitness function value. The fitness value is related by the kind of the objective function, the PSO can be applied to minimize or maximize this function. in this paper we focused in maximizing the objective function in order to improve the results.

Maximize  $f(x_1, x_2) = 21.5 + x_1 \cdot \sin(4\pi x_1) + x_2$   
 $\sin(20\pi x_2)$  (3)  
 here  $-3.0 \leq x_1 \leq 12.1$  and  $4.1 \leq x_2 \leq 5.8$ .

### 5.2. PSO Algorithm [3]

The PSO algorithm depends on its implementation in the following two relations:

$$v_{id} = w \cdot v_{id} + c_1 \cdot r_1 \cdot (p_{best} - v_{id}) + c_2 \cdot r_2 \cdot (g_{best} - v_{id}) \quad (4a)$$

$$x_{id} = x_{id} + v_{id} \quad (4b)$$

where  $c_1$  and  $c_2$  are positive constants,  $r_1$  and  $r_2$  are random functions in the range  $[0,1]$ ,  $p_i = (p_{i1}, p_{i2}, \dots, p_{id})$  represents the  $i$ th particle;  $p_{best} = (p_{best1}, p_{best2}, \dots, p_{bestd})$  represents the best previous position (the position giving the best fitness value) of the  $i$ th particle; the symbol  $g$  represents the index of the best particle among all the particles in the population,  $v_i = (v_{i1}, v_{i2}, \dots, v_{id})$  represents the rate of the position change (velocity) for particle  $i$  [3].

The original procedure for implementing PSO is as follows:

1. Initialize a population of particles with random positions and velocities on  $d$ -dimensions in the problem space.
2. PSO operation includes:
  - a. For each particle, evaluate the desired optimization fitness function in  $d$  variables.
  - b. Compare particle's fitness evaluation with its  $p_{best}$ . If current value is better than  $p_{best}$ , then set  $p_{best}$  equal to the current value, and  $p_{best}$  equals to the current location  $p_i$ .
  - c. Identify the particle in the neighborhood with the best success so far, and assign it index to the variable  $g$ .

d. Change the velocity and position of the particle according to equation (4a) and (4b).

3. Loop to step (2) until a criterion is met.

Like the other evolutionary algorithms, a PSO algorithm is a population based on search algorithm with random initialization, and there is an interaction among population members. Unlike the other evolutionary algorithms, in PSO, each particle flies through the solution space, and has the ability to remember its previous best position, survives from generation to another. The flow chart of PSO algorithm is shown in Figure (1) [11].

### 5.3. The Parameters of PSO [12]

A number of factors will affect the performance of the PSO. These factors are called PSO parameters, these parameters are:

1. Number of particles in the swarm affects the run-time significantly, thus a balance between variety (more particles) and speed (less particles) must be sought.
2. Maximum velocity ( $v_{max}$ ) parameter. This parameter limits the maximum jump that a particle can make in one step.
3. The role of the inertia weight  $w$ , in equation (4a), is considered critical for the PSO's convergence behavior. The inertia weight is employed to control the impact of the previous history of velocities on the current one.
4. The parameters  $c_1$  and  $c_2$ , in equation (4a), are not critical for PSO's convergence. However, proper fine-tuning may result in faster convergence and alleviation of local minima,  $c_1$  than a social parameter  $c_2$  but with  $c_1 + c_2 = 4$ .

5. The parameters  $r_1$  and  $r_2$  are used to maintain the diversity of the population, and they are uniformly distributed in the range  $[0,1]$ .

### 6. Implementation of PSO

In this paper we will try to apply PSO algorithm in optimizing a function. This problem was chosen according to different factors such as representation of the problem (which have a great influence on PSO algorithm) which can be applied more efficiently. Furthermore, this problem has been chosen since it owns a high complexity (the size and the shape of the search space), which cannot be solved using traditional known searches, like exhaustive search method

#### 6.1. Problem Representation

To apply the PSO for maximizing  $f(x)$ , a genetic representation of solution to the problem must be appropriately chosen first. PSO can use the binary representation in which each point is described by a (position), BP (best position) and  $v$  (velocity) vector coded as real values range  $[-1,1]$ . Then when the value of each component is less or equal to 0 it is converted to 0 else it is considered as 1 so it is changed to a binary vector as a string of bits to represent real values of the variable  $x$ , the length of the vector depends on the required precision, which in this example, is six places after the decimal point as we do in representation of each chromosome of GA.

#### 6.2. Initial Population

To optimize the function  $f$  using PSO, we create a population of pop size=10 or 20 particles. All 33 real values converted to 33 corresponding 33-bits which are initialized randomly. for example:

$bp=(-0.31,0.14,0.52,-0.43,-0.78,-0.01,\dots,0.1,-0.32,0.59,0.-72,0.005,0.86)$   
of 33 real components converted to the following binary string:  
 $v = (0,1,1,0,0,0,\dots,1,0,1,0,1,1)$ .

### 6.3. Experimental Results

For this particular problem, we use  $\alpha$  population size  $pop\_size=10$ ,  $c_1=c_2=1$ ,  $\alpha \in [0.4,0.9]$ .  $max=0.5$  and  $vmin=-vmax$ . And  $r_1$  &  $r_2 \in [0,1]$  are chosen randomly with every generation. In Table (2) we provide the vector  $bp$ , generation number for which we noticed the improvement in the evaluation function together with the value of the function, evolution function and values of the variables  $x_j$ .

For this problem, a simulation has been constructed in order to apply PSO, using the parameters mentioned above;

the following results are being obtained after 20 generation from 500 generation:

$vmax=$   
(011011111111111111101101111111111)  
which corresponds to a value  
( $x_1,x_2$ )=(11.990843,5.292858),  
and  $f(vmax)= 35.498874$ .

Our main development in the following section.

### 7. Developing of Applying Real PSO Process

In analytic study of Tables (1) and (2), we noticed that in applying the GA we gain better results from binary PSO with respect to value of evolution function  $f(x_1,x_2)$  ( $f(x_1,x_2)=35.761033$  for GA and  $f(x_1,x_2)=35.498874$  for PSO), but PSO is more better in the number of generation (NG=660 for A while NG=20 for PSO) and the process time. In this section we are developing the application of PSO

algorithm to gain more efficient method to improve the results.

The improvement of applying PSO is as follows:

The component of the vectors  $p$  and  $bp$  are still real and need no change to binary; its just required no more than 2 real variable numbers, these values of  $x_1$  and  $x_2$  say, will considered to be the component of the mentioned vectors.

The new algorithm real PSO is as follows:

#### Algorithm

##### START.

**INPUT:**  $c_1=1$ ,  $pop\_size=10$ ,  $v_{max}=0.5$ ,  $gen.=500$ .

**INITIALIZATION:**  $c_2=c_1$ ,  $v_{min}=-v_{max}$ .

Random real position of all particles.

Random real velocity of all particles.

Computing of fitness values for each particle.

**PROCESS:** For  $i=1$  to  $gen$ .

Evaluating the particles.

Computing of fitness values for each particle.

$Best\_Fit \leftarrow particle\_Fit$ .

END.

**OUTPUT:**  $Best\_Fit$

**END.**

By substituting each value of  $x_1$  and  $x_2$  in equation (1) we can get the evaluation function.

For PSO problem, a simulation has been constructed in order to apply PSO, using the parameters mentioned above; the following results are being obtained after 14 generations:

$(x_1, x_2) = (12.1, 5.8)$ , and  $f(v_{max}) = 35.762019$ .

#### 8. Conclusions

1. The reason that the GA is better than the binary PSO is that the GA gives more varieties in changing a single gene of the chromosome

because of crossover and mutation processes.

2. The real PSO is better than the GA and binary PSO to approach the good fitness result in less process time and in less number of generations.
3. Further suggestions; we can replace the types of binary representation by the real representation when applying PSO.
4. We suggest making a hybrid between GA and PSO in solving optimization of function to improve approaching solution in less time and less number of generations.
5. We suggest using more complicated function including more than two variables to get solutions to these complicated and applicable functions.

#### References

- [1]. Holland, J., "Adaptation in Natural and Artificial Systems", University of Michigan Press., 1975.
- [2]. Koza J. R., "Genetic Programming", 2<sup>nd</sup> Edition, 1992.
- [3]. Shi Y., "Particle Swarm Optimization", Electronic Data Systems, Inc. Kokomo, IN 46902, USA Feature Article, IEEE Neural Networks Society, February 2004.
- [4]. Kennedy J. and Eberhart R. C. "Particle Swarm Optimization", Proceedings of IEEE International Conference on NN, Piscataway, pp. 1942-1948, 1995.
- [5]. Eberhart R. C. and Hu X. "Human Tremor Analysis Using Particle Swarm Optimization" Proceedings of the IEEE Congress on Evolutionary Computation (CEC

- 
- 1999), Washington D.C. pp. 1927-1930, 1999.
- [6]. Mitchell M., “**An Introduction of Genetic Algorithms**”, Abroad Book 1998.
- [7]. Grefenstette J. J., “**Optimization of Control Parameters of Genetic Algorithm**” Genetic Algorithm; IEEE Computer Society, 1992.
- [8]. Michalewicz, Zbigniew, “**Genetic Algorithms + Data Structures = Evolution Programs**”, Springer-Verlag, 1999.
- [9]. Pomeroy P. “**An Introduction to Particle Swarm Optimization**”, Article, March, [www.adaptiveview.com](http://www.adaptiveview.com), page 1-7, 2003.
- [10]. Ribeiro P. F. and Kyle W. S., “**A Hybrid Particle Swarm and Neural Network Approach for Reactive Power Control**”, Member, 2003. <http://engr.calvin.edu/WEBPAGE/.../engir302/Reactivepower-PSO-wks.pdf>
- [11]. Zhou Y., and et al, “**Particle Swarm Optimization Based Approach for Optical Finite Impulse Response Filter Design**”, Optical Society of America, 2003.
- [12]. Parsopoulos K. E. and Vrahatis M.N., “**Recent Approaches to Global Optimization Problems through Particle Swarm Optimization**”, Kluwer Academic Publishers, Netherlands, Natural Computing 1, pp 235–306, 2002.



Table (1) Results of 1000 generations for optimization of a function problem

Chromosome	Gen. No.	Evolution Function	Variables $x_i$	
			$x_1$	$x_2$
010001001011010000010001001011010	0	26.566769	2.334464	4.699438
011100101001111110011100101001111	0	28.068086	4.451298	5.711127
110101010111100111110101010111100	0	33.236484	10.664784	4.507321
11110110000111011111101100001110	0	34.012620	11.044728	4.849531
000110011100101111000110011100101	0	34.755134	11.445236	5.210263
010100100110101111010100100110101	1	34.823983	11.484981	5.246061
111110111110101111111110111110101	1	34.863961	11.508310	5.267073
110000101111011111110000101111011	2	35.415673	11.853232	5.577739
000110101111011111000110101111011	3	35.417473	11.854442	5.578829
011111001101111111011111001101111	4	35.667943	12.029898	5.736860
000010111011111111000010111011111	9	35.719240	12.067800	5.770998
011101111011111111011101111011111	25	35.721555	12.069528	5.772554
101101010111111111101101010111111	34	35.736241	12.080530	5.782464
001111011111111111001111011111111	100	35.756931	12.096140	5.796523
110000111111111111110000111111111	296	35.757463	12.096543	5.796887
001100111111111111001100111111111	530	35.758147	12.097062	5.797354
101100111111111111101100111111111	637	35.758223	12.097119	5.797405
101110111111111111101110111111111	638	35.759438	12.098041	5.798236
011001111111111111101100111111111	642	35.760122	12.098559	5.798702
010011111111111111010011111111111	660	35.761033	12.099251	5.799325

**Table (2) Results of 500 generations for optimization of a function using binary PSO.**

Corresponding string of bp vector	Gen. No.	Evolution Function	Variables $x_i$	
			$x_1$	$x_2$
1010 11101011100101101001011101011		26. 585720	2.3 35744	4. 720606
1101 01000111011111100100011110110		29. 384191	6.4 21061	5. 314856
1011 00100011101010101110111100111		32. 639301	9.5 89755	5. 577428
0110 01110000011010111101000110110		34. 510307	11. 48152	5. 573824
0110 01001111110110111100101000000		34. 584349	11. 568385	4. 738557
0111 01001110110110111100001000000		35. 041305	11. 561016	5. 588375
1001 00110000110110111111101111101		34. 854352	11. 555141	5. 135502
1001 01101010011001111111010101110	0	35. 050233	11. 721611	5. 104995
1001 00100100011011111101010010111	2	35. 380306	11. 954669	5. 153089
0111 10010100000011111101010011000	7	35. 436987	11. 866480	5. 598025
1101 00011001000111111100001011100	0	35. 498874	11. 990843	5. 292858

**Table (3) the results of applying real PSO of optimization to the function.**

Gen. No.	Evolution Function	Variables $x_i$	
		$x_1$	$x_2$
0	30.106759	7.719931	4.649781
0	31.722201	8.813922	5.395647
1	32.061658	9.1616653	5.295070
5	33.108293	10.006154	5.515154
5	33.124616	10.237736	4.983999
6	33.329142	10.442914	4.924686
6	33.521591	10.506154	5.096958
7	33.795334	10.956004	4.720708
8	34.152473	11.164021	4.837212
9	34.661042	11.592390	4.793219
11	35.544439	12.100000	5.109219
12	35.635955	12.100000	5.280910
13	35.707642	12.100000	5.465228
14	35.762019	12.100000	5.800000

Table (4) Shows a comparison with the three Algorithms (Bin.GA, Bin. PSO and Real PSO).

Algorithm	Gen. No.	Evolution Function	Variables	
			$x_1$	$x_2$
Bin. GA	660	35.761033	12.099251	5.799325
Bin. PSO	20	35. 498874	11. 990843	5. 292858
Real PSO	14	35.762019	12.100000	5.800000

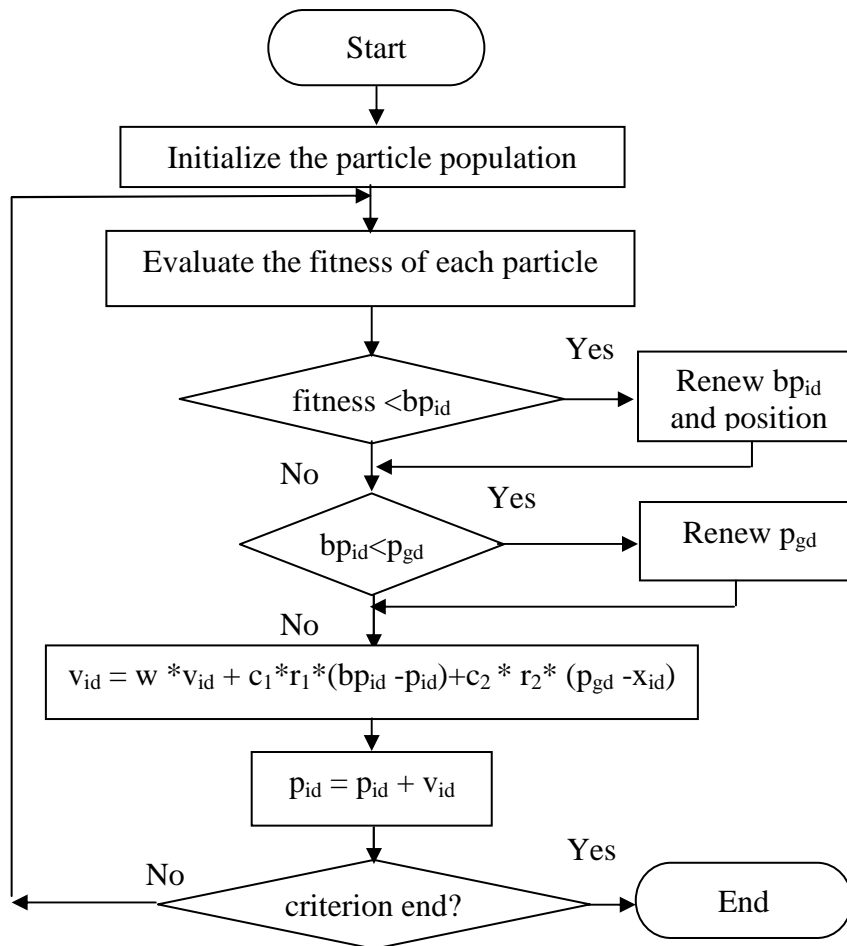


Figure (1) Flowchart of PSO Algorithm.