20

Particle Swarm Optimization

Adam Slowik
Department of Electronics and Computer Science
Koszalin University of Technology, Koszalin, Poland

CONTENTS

20.1 Introduction 265
20.2 Original PSO algorithm 266
20.2.1 Pseudo-code of global version of PSO algorithm 266
20.2.2 Description of the global version of the PSO algorithm 267
20.2.3 Description of the local version of the PSO algorithm .. 268
20.3 Source-code of global version of PSO algorithm in Matlab 269
20.4 Source-code of global version of PSO algorithm in C++ 270
20.5 Step-by-step numerical example of global version of PSO algorithm 272
20.6 Conclusions 276
References 277

20.1 Introduction

In 1995, in the paper [1] by Kennedy et al., the first version of a particle swarm optimization (PSO) algorithm was proposed. The main inspiration for the PSO algorithm was the social behavior of such organisms as birds (a bird flock) or fish (a fish school). The PSO [1] and ACO [2] algorithms can be considered as the fathers of the new research area named swarm intelligence. The PSO algorithm is a global optimization algorithm which was originally developed for solving problems where the solutions are represented by the points in a multidimensional search space of parameters with continuous values (algorithm for real-valued optimization). Due to the PSO algorithm we can also obtain solutions for non-differentiable problems which may be irregular, dynamically changing with time, or noisy. Up until now many modifications of the PSO algorithm have been elaborated. Among them are, for example: the micro-PSO for a high dimensional optimization problem (it operates on a swarm consisting of a small number of particles) [5], discrete PSO (the

combinatorial problem can be solved using this algorithm) [6], correlation-based binary PSO [7], heterogeneous strategy PSO (the proportion of particles adopts a fully informed strategy to enhance the converging speed while the rest is singly informed to maintain the diversity) [8], and multi-objective PSO [9]. The rest of this paper is organized as follows: in Section 20.2 the pseudo-code for the global version of the PSO algorithm is presented and discussed (also a short introduction to the local version of PSO algorithm is given), in Sections 20.3 and 20.4 the source-codes for the global version of the PSO algorithm are shown in Matlab and C++ programming language respectively, in Section 20.5 we present in detail the numerical example of the global version of the PSO algorithm, and in Section 20.6 some conclusions are provided.

20.2 Original PSO algorithm

20.2.1 Pseudo-code of global version of PSO algorithm

The pseudo-code for the global version of the PSO algorithm is presented in Algorithm 18.

Algorithm 18 Pseudo-code of the global version PSO.

determine the $D-th$ dimensional objective function $OF(.)$
determine the range of variability for each $j-th$ dimension $\left[P_{i,j}^{min}, P_{i,j}^{max}\right]$
determine the PSO algorithm parameter values such as N – number of particles in the swarm, c_1, c_2 – learning factor
randomly create swarm P which consists of N particles (each particle is a D-dimensional vector)
create the D-dimensional $Gbest$ vector
for each i-th particle P_i from swarm P **do**
    create a $Pbest_i$ particle equal to i-th particle P_i
    create a velocity V_i for each particle P_i
    evaluate the particle P_i using $OF(.)$ function
end for
assign the best particle P_i to the $Gbest$
while termination condition not met **do**
    for each i particle in the swarm P **do**
        for each j dimension **do**
            update the velocity $V_{i,j}$ using formula
            $V_{i,j} = V_{i,j} + c_1 \cdot r_1 \cdot (Pbest_{i,j} - P_{i,j}) + c_2 \cdot r_2 \cdot (Gbest_j - P_{i,j})$
            update the particle $P_{i,j}$ using formula
            $P_{i,j} = P_{i,j} + V_{i,j}$
            if $P_{i,j} < P_j^{min}$ **then** $P_{i,j} = P_j^{min}$
            end if
            if $P_{i,j} > P_j^{max}$ **then** $P_{i,j} = P_j^{max}$

22: **end if**
23: **end for**
24: evaluate the particle P_i using $OF(.)$ function
25: **if** $OF(P_i)$ is better than $OF(Pbest_i)$ **then**
26: assign the particle P_i to the particle $Pbest_i$
27: **end if**
28: **end for**
29: select the best particle from swarm P and assign it to particle T
30: **if** $OF(T)$ is better than $OF(Gbest)$ **then**
31: assign the particle T to the particle $Gbest$
32: **end if**
33: **end while**
34: return the $Gbest$ as a result

20.2.2 Description of the global version of the PSO algorithm

In Algorithm 18 the pseudo-code of the global version of the PSO algorithm was presented. Now in this section we will discuss this algorithm in detail. Before we start our PSO algorithm we should have defined an objective function $OF(.)$ (step 1), and all the algorithm parameters such as learning coefficient c_1 and c_2, and number of particles N (step 2). In literature, we can find that the summation of c_1 and c_2 should be less than or equal to 4. Next, we randomly create the swarm P which consists of N particles (step 3). Each particle is represented by the D-dimensional vector, and its particular elements represent the decision variables from the $OF(.)$. Of course each j-th decision variable must be constrained by the lower P_j^{min} and upper P_j^{max} bound value. Therefore the particle P_i is represented by the following vector:

$$P_i = \{P_{i,1}, P_{i,2}, P_{i,3}, ..., P_{i,D-1}, P_{i,D}\}$$
where: $P_{i,1}, P_{i,2}$, and so on, are the values from the range $\left[P_j^{min}, P_j^{max}\right]$.

In step 4, the $Gbest$ vector is created. In this vector the best solution found so far is stored. At the start, the $Gbest$ is a D-dimensional zero vector. Next, the $Pbest_i$ particle is created for each corresponding P_i particle. At the start (step 6), the $Pbest_i$ particle is equal to the P_i particle. In the $Pbest_i$ particle the best particle P_i which was found so far will be stored. In the 7th step, the velocity vector V_i is created for particle P_i. The newly created vector V_i consists of only zero values. When the first swarm P is created, we can evaluate the quality of each particle P_i using the previously defined $OF(.)$ function (step 8). After swarm evaluation, we can select, from the swarm P, the best particle P_i which will be stored in the particle $Gbest$ (step 9). By "the best" we mean the particle with the lowest value of $OF(.)$ function (for the minimization tasks) or with the highest value of $OF(.)$ function (for the maximization

tasks). In the 11th step, the main algorithm loop is started until the termination criteria are fulfilled. As a termination criterion we can take into account, for example, the number of generations, the convergence of the algorithm, or the computational time. In the main algorithm loop, for all particles in swarm P, we update each j-th element value from the velocity vector V_i which is assigned to the particle P_i (step 15), and we update each j-th element value from the particle P_i using values from the previously computed velocity vector V_i (step 17). When the particle P_i is updated, we evaluate the quality of this particle using the $OF(.)$ function (step 19). Next, in step 20, if the value of $OF(P_i)$ is lower than the value of $OF(Pbest_i)$ (for the minimization tasks) or higher than the value of $OF(Pbest_i)$ then the particle P_i will be assigned to the corresponding particle $Pbest_i$ (step 21). When all particles from swarm P have been evaluated then the best particle is selected from the swarm and assigned to the temporary particle T (step 24). If the $OF(.)$ value for particle T is lower than the $OF(.)$ value for particle $Gbest$ (for minimization tasks) or higher than the $OF(.)$ value for particle $Gbest$ (for maximization tasks) (step 25) then the particle T will be assigned to the particle $Gbest$ (step 26). The algorithm (steps 11 to 28) is repeated until its termination criterion is satisfied. When the termination criterion is fulfilled, the particle $Gbest$ is returned as an algorithm result. In practical realization of the PSO algorithm the formula from the 16th step of the algorithm (see Algorithm 18) is very often replaced by the following expression:

$$V_{i,j} = \omega \cdot V_{i,j} + c_1 \cdot r_1 \cdot (Pbest_{i,j} - P_{i,j}) + c_2 \cdot r_2 \cdot (Gbest_j - P_{i,j}) \quad (20.1)$$

where ω is a factor called an inertia weight which significantly affects the convergence and exploration-exploitation trade-off in PSO, the typical value for inertia weight factor is from the range $[0.4, 0.9]$ [3].

20.2.3 Description of the local version of the PSO algorithm

In the global version of the PSO algorithm each particle goes toward the best particle in the whole swarm. In the local version of the PSO algorithm each particle goes toward the best particle in its neighborhood. Therefore the formula from the 16th step of the algorithm (see Algorithm 18) is replaced by the following expression (without inertia weight):

$$V_{i,j} = V_{i,j} + c_1 \cdot r_1 \cdot (Pbest_{i,j} - P_{i,j}) + c_2 \cdot r_2 \cdot (Lbest_{i,j} - P_{i,j}) \quad (20.2)$$

or by the expression with inertia weight factor:

$$V_{i,j} = \omega \cdot V_{i,j} + c_1 \cdot r_1 \cdot (Pbest_{i,j} - P_{i,j}) + c_2 \cdot r_2 \cdot (Lbest_{i,j} - P_{i,j}) \quad (20.3)$$

where in both expressions 20.2 and 20.3 the $Lbest_i$ is the best particle in the neighborhood of particle P_i. The neighborhood can be a geometrical one, or a social one. When the neighborhood is geometrical, we can compute the

distances between all particles in the swarm (for example using Euclidean distance) and then for each particle P_i we select the best particle $Lbest_i$ among the m nearest neighborhoods of particle P_i. The social neighborhood is defined on the basis of a topology which depends only on the particle index i [4]. In practical application, the neighborhood of each particle is defined (at the start of the PSO algorithm) as a list of particles which does not change with the iterations. The topology which is more commonly used is a ring with a $k = 2$ (each particle has two neighbors – first on the left side, and second on the right side). In ring topology the left ($Left_i$) and right ($Right_i$) neighbor for the $i - th$ particle P_i can be determined using the formula:

$$Left_i = \begin{cases} i+1 & \text{if } i < N \\ 1 & \text{if } i = N \end{cases} \tag{20.4}$$

$$Right_i = \begin{cases} i-1 & \text{if } i > 1 \\ N & \text{if } i = 1 \end{cases} \tag{20.5}$$

Next, the best particle (from both the particles, $Left_i$ and $Right_i$) is selected as a $Lbest_i$ particle for particle P_i. The advantages of the local version of the PSO algorithm are that it may search a different area of the solutions space or explore different local optima.

20.3 Source-code of global version of PSO algorithm in Matlab

In Listing 20.1 the source-code for the objective function which will optimize by the PSO algorithm is shown. In the function $OF(.)$, the input parameter is a whole swarm P. The result of $OF(.)$ function is an N-dimensional column vector with the objective function values for each particle P_i from the swarm P. The objective function is given by formula 20.6. The PSO algorithm minimizes this objective function.

$$OF(P_i) = \sum_{j=1}^{D} P_{i,j}^2 \qquad \text{where } -5.12 \leqslant P_{i,j} \leqslant 5.12 \tag{20.6}$$

```
function [out]=OF(P)
[x,y]=size(P);
out=zeros(x,1);
for i=1:x
    for j=1:y
    out(i,1)=out(i,1)+P(i,j)^2;
    end
end
```

Listing 20.1
Definition of objective function $OF(.)$ in Matlab.

```
% declaration of the parameters of PSO algorithm
N=100; c1=2; c2=2; MAX=100; Iter=0; D=20;
% constraints definition for all decision variables
Pmin(1,1:D)=-5.12; Pmax(1,1:D)=5.12;
% the swarm P is created randomly
P=zeros(N,D)+(Pmax(1,:)-Pmin(1,:)).*rand(N,D)+Pmin(1,:);
% vector Gbest} is created
Gbest=zeros(1,D);
% all the particles from swarm P are remembered in swarm Pbest
Pbest=P;
% velocity vector is created for all particles in swarm P
V=zeros(N,D);
% the whole swarm P is evaluated using objective function OF(.)
Eval=OF(P);
% the swarm Pbest is evaluated using objective function OF(.)
EvalPbest(:,1)=Eval(:,1);
% the best particle is chosen from the swarm P
[Y,I]=min(Eval(:,1));
% the best particle from the swarm P is stored in Gbest
Gbest(1,:)=P(I,:);
% the OF(.) value of the best particle is stored in EvalGbest
EvalGbest=Y;
% main program loop starts
while(Iter<MAX)
% number of iteration is increased by one
    Iter=Iter+1;
% the velocity vector for each particle is computed
    V=V+c1*rand()*(Pbest-P)+c2*rand()*(Gbest-P);
% the new position for each particle is computed
    P=P+V;
% the constraints for each decision variable are checked
    for i=1:N
        for j=1:D
            if P(i,j)<Pmin(1,j)
                P(i,j)=Pmin(1,j);
            end
            if P(i,j)>Pmax(1,j)
                P(i,j)=Pmax(1,j);
            end
        end
    end
% the whole swarm P is evaluated using objective function OF(.)
    Eval=OF(P);
% the Pbest swarm is updated if needed
    for i=1:N
        if Eval(i,1)<EvalPbest(i,1)
            Pbest(i,:)=P(i,:);
        end
    end
% the best particle is chosen from the swarm P
    [Y,I]=min(Eval(:,1));
% the Gbest vector is updated if needed
    if Eval(I,1)<EvalGbest
        Gbest(1,:)=P(I,:);
        EvalGbest=Y;
    end
end
% the result of PSO algorithm is returned
disp(EvalGbest);
```

Listing 20.2
Source-code of the global version of the PSO in Matlab.

20.4 Source-code of global version of PSO algorithm in C++

```
#include <iostream>
#include <time.h>
using namespace std;
// definition of the objective function OF(.)
float OF(float x[], int size_array)
{
    float t=0;
    for(int i=0; i<size_array; i++)
    {
        t=t+x[i]*x[i];
    }
    return t;
}
// generate pseudo random values from the range [0, 1)
float r()
{
    return (float)(rand()%1000)/1000;
}
// main program function
int main()
{
    srand(time(NULL));
// initialization of the parameters
    int N=100; int MAX=100; int Iter=0; int D=20;
    int i,j,k;
    int TheBest=0;
    float c1=2; float c2=2;
    float Pmin[D]; float Pmax[D];
// initialization of the constraints
    for (int j=0; j<D; j++)
    {
        Pmin[j]=-5.12; Pmax[j]=5.12;
    }
// initialization of all data structures which are needed by PSO
    float P[N][D]; float Pbest[N][D];
    float V[N][D]; float Eval[N]; float EvalPbest[N];
    float Gbest[D]; float EvalGbest;
// randomly create swarm P and Pbest; create velocity vectors V
    for(int i=0; i<N; i++)
    {
        for(int j=0; j<D; j++)
        {
        P[i][j]=(Pmax[j]-Pmin[j])*r()+Pmin[j];
        Pbest[i][j]=P[i][j];
        V[i][j]=0;
        }
// evaluate all the particles in the swarm P
        Eval[i]=OF(P[i],D);
        EvalPbest[i]=Eval[i];
// find the best particle in the swarm P
        if (Eval[i]<Eval[TheBest]) TheBest=i;
    }
// assign the best particle to Gbest
    for(j=0; j<D; j++) Gbest[j]=P[TheBest][j];
    EvalGbest=Eval[TheBest];
// main program loop
    while(Iter<MAX)
    {
        Iter++;
        TheBest=0;
```

```
61          for(int i=0; i<N; i++)
62          {
63              for(int j=0; j<D; j++)
64              {
65 // for each particle compute the velocity vector
66                  V[i][j]=V[i][j]+c1*r()*(Pbest[i][j]-P[i][j])+c2*r()*(
       Gbest[j]-P[i][j]);
67 // update each particle using velocity vector
68                  P[i][j]=P[i][j]+V[i][j];
69 // check the constraints for each dimension
70                  if (P[i][j]<Pmin[j]) P[i][j]=Pmin[j];
71                  if (P[i][j]>Pmax[j]) P[i][j]=Pmax[j];
72              }
73 // evaluate each particle from the swarm P
74              Eval[i]=OF(P[i],D);
75              if (Eval[i]<EvalPbest[i])
76              {
77 // for each particle update its Pbest vector
78                  for(int j=0; j<D; j++) Pbest[i][j]=P[i][j];
79              }
80              if (Eval[i]<Eval[TheBest]) TheBest=i;
81          }
82 // update Gbest vector if better particle is found
83          if (Eval[TheBest]<EvalGbest)
84          {
85              for(int j=0; j<D; j++) Gbest[j]=P[TheBest][j];
86              EvalGbest=Eval[TheBest];
87          }}
88      cout << "EvalGbest = " << EvalGbest << endl;
89      getchar();
90      return 0;}
```

Listing 20.3
Source-code of the global version of the PSO in C++.

20.5 Step-by-step numerical example of global version of PSO algorithm

In the first step, let's assume that we want to minimize a mathematical function given by equation 20.6 where D is equal to 5. In the second step, we determine the PSO algorithm parameter values such as $N = 6$, $c_1 = 2$, and $c_2 = 2$. In the third step, the swarm P which consists of 6 particles (each particle is a 5-dimensional vector) is created.
$P_1 = \{-2.956, 2.622, -5.118, -1.737, 1.693\}$
$P_2 = \{1.314, 3.581, 1.902, 3.873, -4.420\}$
$P_3 = \{0.623, 1.662, 2.318, -3.087, 0.453\}$
$P_4 = \{-2.743, -2.752, -2.903, 3.926, 1.562\}$
$P_5 = \{-1.970, 4.433, -2.922, -1.918, -1.417\}$
$P_6 = \{-2.127, 0.680, -0.177, -1.718, 0.957\}$

In the fourth step, the 5-dimensional *Gbest* vector is created.
$Gbest = \{0, 0, 0, 0, 0\}$

In the fifth step, the $Pbest_i$ particle is created for each i-th particle P_i. At the start, the particles $Pbest_i$ are the same as particles P_i.

$Pbest_1 = \{-2.956, 2.622, -5.118, -1.737, 1.693\}$
$Pbest_2 = \{1.314, 3.581, 1.902, 3.873, -4.420\}$
$Pbest_3 = \{0.623, 1.662, 2.318, -3.087, 0.453\}$
$Pbest_4 = \{-2.743, -2.752, -2.903, 3.926, 1.562\}$
$Pbest_5 = \{-1.970, 4.433, -2.922, -1.918, -1.417\}$
$Pbest_6 = \{-2.127, 0.680, -0.177, -1.718, 0.957\}$

In the sixth step, for each particle P_i the 5-dimensional velocity vector V_i is created. At the start each V_i vector consists of zeros.
$V_1 = \{0, 0, 0, 0, 0\}$
$V_2 = \{0, 0, 0, 0, 0\}$
$V_3 = \{0, 0, 0, 0, 0\}$
$V_4 = \{0, 0, 0, 0, 0\}$
$V_5 = \{0, 0, 0, 0, 0\}$
$V_6 = \{0, 0, 0, 0, 0\}$

In the seventh step, we evaluate each particle P_i using the objective function OF(.) given by equation 20.6. For particle $P_1 = \{-2.956, 2.622, -5.118, -1.737, 1.693\}$ we obtain:

$OF(P_1) = \sum_{j=1}^{5} P_{1,j}^2 = (-2.956)^2 + (2.622)^2 + (-5.118)^2 + (-1.737)^2 + (1.693)^2 = 8.738 + 6.875 + 26.194 + 3.017 + 2.866 = 47.690$

If we repeat the same computations for all particles in the swarm P, we obtain the $Eval\_P_i$ values which are the following:
$Eval\_P_1 = OF(P_1) = 47.690$
$Eval\_P_2 = OF(P_2) = 52.704$
$Eval\_P_3 = OF(P_3) = 18.258$
$Eval\_P_4 = OF(P_4) = 41.378$
$Eval\_P_5 = OF(P_5) = 37.757$
$Eval\_P_6 = OF(P_6) = 8.885$

In the eighth step, we take the best particle P_i (with the smallest value of objective function – our goal is OF(.) minimization) and assign it to the *Gbest*. In our case, the best particle is P_6, therefore $Gbest = P_6 = \{-2.127, 0.680, -0.177, -1.718, 0.957\}$ and of course $OF(Gbest) = 8.885$.

In the ninth step, the main loop of the algorithm starts. We check whether the algorithm termination condition is fulfilled. If yes we jump to the sixteenth step. If no we go to the tenth step.

In the tenth step, the velocity vector V_i is updated for each particle P_i according to the formula:
$V_{i,j} = V_{i,j} + c_1 \cdot r_1 \cdot (Pbest_{i,j} - P_{i,j}) + c_2 \cdot r_2 \cdot (Gbest_j - P_{i,j})$

Now, we show in detail how the velocity vector V_1 is computed for single particle P_1.

$V_{1,1} = V_{1,1} + c_1 \cdot r_1 \cdot (Pbest_{1,1} - P_{1,1}) + c_2 \cdot r_2 \cdot (Gbest_1 - P_{1,1})$
Two numbers r_1 and r_2 from the range [0; 1] are randomly chosen for computation of $V_{1,1}$. Let us assume that $r_1 = 0.89$ and $r_2 = 0.53$ have been randomly selected for $V_{1,1}$ computation.
$V_{1,1} = 0 + 2 \cdot 0.89 \cdot (-2.956 - (-2.956)) + 2 \cdot 0.53 \cdot (-2.127 - (-2.956))$
$V_{1,1} = 0 + 2 \cdot 0.89 \cdot 0 + 2 \cdot 0.53 \cdot 0.829 = 0.879$

$V_{1,2} = V_{1,2} + c_1 \cdot r_1 \cdot (Pbest_{1,2} - P_{1,2}) + c_2 \cdot r_2 \cdot (Gbest_2 - P_{1,2})$
Two numbers r_1 and r_2 from the range [0; 1] are randomly chosen for computation of $V_{1,2}$. Let us assume that $r_1 = 0.21$ and $r_2 = 0.75$ have been randomly selected for $V_{1,2}$ computation.
$V_{1,2} = 0 + 2 \cdot 0.21 \cdot (2.622 - 2.622) + 2 \cdot 0.75 \cdot (0.680 - 2.622)$
$V_{1,2} = 0 + 2 \cdot 0.21 \cdot 0 + 2 \cdot 0.75 \cdot (-1.942) = 2.913$

$V_{1,3} = V_{1,3} + c_1 \cdot r_1 \cdot (Pbest_{1,3} - P_{1,3}) + c_2 \cdot r_2 \cdot (Gbest_3 - P_{1,3})$
Two numbers r_1 and r_2 from the range [0; 1] are randomly chosen for computation of $V_{1,3}$. Let us assume that $r_1 = 0.33$ and $r_2 = 0.66$ have been randomly selected for $V_{1,3}$ computation.
$V_{1,3} = 0 + 2 \cdot 0.33 \cdot (-5.118 - (-5.118)) + 2 \cdot 0.66 \cdot (-0.177 - (-5.118))$
$V_{1,3} = 0 + 2 \cdot 0.33 \cdot 0 + 2 \cdot 0.66 \cdot 4.941 = 6.522$

$V_{1,4} = V_{1,4} + c_1 \cdot r_1 \cdot (Pbest_{1,4} - P_{1,4}) + c_2 \cdot r_2 \cdot (Gbest_4 - P_{1,4})$
Two numbers r_1 and r_2 from the range [0; 1] are randomly chosen for computation of $V_{1,4}$. Let us assume that $r_1 = 0.63$ and $r_2 = 0.85$ have been randomly selected for $V_{1,4}$ computation.
$V_{1,4} = 0 + 2 \cdot 0.63 \cdot (-1.737 - (-1.737)) + 2 \cdot 0.85 \cdot (-1.718 - (-1.737))$
$V_{1,4} = 0 + 2 \cdot 0.63 \cdot 0 + 2 \cdot 0.85 \cdot 0.019 = 0.032$

$V_{1,5} = V_{1,5} + c_1 \cdot r_1 \cdot (Pbest_{1,5} - P_{1,5}) + c_2 \cdot r_2 \cdot (Gbest_5 - P_{1,5})$
Two numbers r_1 and r_2 from the range [0; 1] are randomly chosen for computation of $V_{1,5}$. Let us assume that $r_1 = 0.68$ and $r_2 = 0.88$ have been randomly selected for $V_{1,5}$ computation.
$V_{1,5} = 0 + 2 \cdot 0.68 \cdot (1.693 - 1.693) + 2 \cdot 0.88 \cdot (0.957 - 1.693)$
$V_{1,5} = 0 + 2 \cdot 0.68 \cdot 0 + 2 \cdot 0.88 \cdot (-0.736) = -1.295$

After computations the velocity vector V_1 is as follows:
$V_1 = \{0.879, 2.913, 6.522, 0.032, -1.295\}$

If we do the same computations for the other velocity (V_2, V_3, V_4, V_5, and V_6) vectors we obtain the following results:
$V_2 = \{-3.832, -7.923, -2.693, -8.597, 7.839\}$
$V_3 = \{-3.498, -1.981, -4.954, 5.790, -0.170\}$

$V_4 = \{2.621, 6.047, 1.842, -7.112, -3.196\}$
$V_5 = \{1.472, -8.642, 4.252, 2.536, 4.918\}$
$V_6 = \{1.486, -1.254, 0.122, 2.522, -0.956\}$

In the eleventh step, we update each particle P_i using the velocity vector V_i corresponding to it ($P_{i,j} = P_{i,j} + V_{i,j}$).
Now, we show in detail how the particle vector P_1 is computed using velocity vector V_1.
$P_{1,1} = P_{1,1} + V_{1,1} = -2.956 + 0.879 = -2.077$
$P_{1,2} = P_{1,2} + V_{1,2} = 2.622 + 2.913 = 5.535$
$P_{1,3} = P_{1,3} + V_{1,3} = -5.118 + 6.522 = 1.404$
$P_{1,4} = P_{1,4} + V_{1,4} = -1.737 + 0.032 = -1.705$
$P_{1,5} = P_{1,5} + V_{1,5} = 1.693 + (-1.295) = 0.398$

So, the new vector for particle P_1 is as follows:
$P_1 = \{-2.077, 5.535, 1.404, -1.705, 0.398\}$

Now, we should check whether all constraints are satisfied for each decision variable from particle P_1. As we know, the constraint for each variable is $P_{i,j} = [P_j^{min}, P_j^{max}] = [-5.12, 5.12]$. We can see that the value for the second decision variable in particle P_1 does not satisfy the constraint ($P_{i,j} > P_j^{max}; 5.535 > 5.12$). Therefore the value of the second variable for particle P_i is set up to value $P_j^{max} = 5.12$, and the new vector for particle P_1 is as follows:
$P_1 = \{-2.077, 5.12, 1.404, -1.705, 0.398\}$

If we do the same computations for the other particles (P_2, P_3, P_4, P_5, and P_6) we obtain the following results:
$P_2 = \{-2.518, -4.342, -0.791, -4.724, 3.419\}$
$P_3 = \{-2.875, -0.319, -2.636, 2.703, 0.283\}$
$P_4 = \{-0.122, 3.295, -1.061, -3.186, -1.634\}$
$P_5 = \{-0.498, -4.209, 1.33, 0.618, 3.501\}$
$P_6 = \{-0.641, -0.574, -0.055, 0.804, 0.001\}$

In the twelfth step, we evaluate all the particles P_i from the swarm P using objective function $OF(.)$ given by the equation 20.6.
For particle $P_1 = \{-2.077, 5.12, 1.404, -1.705, 0.398\}$ we obtain:

$OF(P_1) = \sum_{j=1}^{5} P_{1,j}^2 = (-2.077)^2 + (5.12)^2 + (1.404)^2 + (-1.705)^2 + (0.398)^2 = 4.314 + 26.214 + 1.971 + 2.907 + 0.158 = 35.564$

If we repeat the same computations for all particles in the swarm P, we obtain the $Eval\_P_i$ values which are the following:
$Eval\_P_1 = OF(P_1) = 35.564$
$Eval\_P_2 = OF(P_2) = 59.825$
$Eval\_P_3 = OF(P_3) = 22.702$

$Eval\_P_4 = OF(P_4) = 24.818$
$Eval\_P_5 = OF(P_5) = 32.371$
$Eval\_P_6 = OF(P_6) = 1.390$

In the thirteenth step, we compare the value of the objective function for particle P_i with the value of the objective function for particle $Pbest_i$ corresponding to it. When the value of the objective function for particle P_i is smaller than the value of the objective function for particle $Pbest_i$ then the particle P_i is written down to the particle $Pbest_i$.
$OF(P_1) = 35.564 < OF(Pbest_1) = 47.690$ then $Pbest_1$ is updated by P_1
$OF(P_2) = 59.825 > OF(Pbest_2) = 52.704$ then $Pbest_2$ is not updated
$OF(P_3) = 22.702 > OF(Pbest_3) = 18.258$ then $Pbest_3$ is not updated
$OF(P_4) = 24.818 < OF(Pbest_4) = 41.378$ then $Pbest_4$ is updated by P_4
$OF(P_5) = 32.371 < OF(Pbest_5) = 37.757$ then $Pbest_5$ is updated by P_5
$OF(P_6) = 1.390 < OF(Pbest_6) = 8.885$ then $Pbest_6$ is updated by P_6

Therefore after this operation, we have a new set of *Pbest* particles which is as follows:
$Pbest_1 = \{-2.077, 5.12, 1.404, -1.705, 0.398\}$
$Pbest_2 = \{1.314, 3.581, 1.902, 3.873, -4.420\}$
$Pbest_3 = \{0.623, 1.662, 2.318, -3.087, 0.453\}$
$Pbest_4 = \{-0.122, 3.295, -1.061, -3.186, -1.634\}$
$Pbest_5 = \{-0.498, -4.209, 1.33, 0.618, 3.501\}$
$Pbest_6 = \{-0.641, -0.574, -0.055, 0.804, 0.001\}$

In the fourteenth step, we select the best particle from swarm P. We can see that the best particle is particle P_6 (objective function $OF(P_6) = 1.390$). After this, we check whether the $OF(P_6)$ is better than $OF(Gbest)$.
$OF(P_6) = 1.390 < OF(Gbest) = 8.885$ then *Gbest* is updated by P_6.

After this operation the new vector *Gbest* is the following:
$Gbest = \{-0.641, -0.574, -0.055, 0.804, 0.001\}$

In the fifteenth step we return to the ninth step.

In the sixteenth step, we return the particle *Gbest* as a result of the algorithm operation, and we stop the algorithm.

20.6 Conclusions

In this chapter we have aimed to show the main principles of the PSO algorithm. We have shown how this algorithm works (in both the global and local

version). In addition, we have provided source-codes in Matlab and in C++ programming language which could help with others' implementation of this algorithm. Finally, the step-by-step numerical example of the PSO algorithm has been presented in detail for a better understanding of the particular operations which occur in the PSO algorithm. We believe that this chapter will make the implementation of one's own PSO algorithm in any programming language easier for anybody to achieve.

References

1. J. Kennedy, R.C. Eberhart. "Particle swarm optimization" in *Proc. of IEEE International Conference on Neural Networks*, 1995, pp. 1942-1948.
2. A. Colorni, M. Dorigo, V. Maniezzo. "Distributed optimization by ant colonies" in *Proc. of the European Conf. on Artificial Life*, 1991, pp. 134-142.
3. J. Xin, G. Chen, Y. Hai. "A particle swarm optimizer with multi-stage linearly-decreasing inertia weight" in Proc. of International Joint Conference on Computational Sciences and Optimization, 2009, pp. 505-508.
4. F. Marini, B. Walczak. "Particle swarm optimization (PSO). A tutorial". Chemometrics and Intelligent Laboratory Systems, vol. 149, part B, pp. 153-165, 2015.
5. W.H. Han. "A new simple micro-PSO for high dimensional optimization problem". Applied Mechanics and Materials, vol. 236-237, pp. 1195-1200, 2012.
6. Z. Wen-Liang, G.Z. Jun, C. Wei-Neng. "A novel discrete particle swarm optimization to solve traveling salesman problem" in *IEEE Congress on Evolutionary Computation*, 2007, pp. 3283-3287.
7. H.J. Wang, R.M. Ke, J.H. Li, Y. An, K. Wang, L. Yu. "A correlation-based binary particle swarm optimization method for feature selection in human activity recognition". *International Journal of Distributed Sensor Networks*, vol. 14(4), 2018
8. W.B. Du, W. Ying, G. Yan, Y.B. Zhu, X.B. Cao. "Heterogeneous strategy particle swarm optimization". *IEEE Transactions on Circuits and Systems II - Express Briefs*, vol. 64(4), pp. 467-471, 2017.
9. K. Sethanan, W. Neungmatcha. "Multi-objective particle swarm optimization for mechanical harvester route planning of sugarcane fields operations". *European Journal of Operational Research*, vol. 252(3), pp. 969-984, 2016.