Hacettepe University - Computer Engineering

Swarm Systems CMP 756

Homework-3: Particle Swarm Optimization

Student Name Ayça KULA
Student ID: 202237285

Spring 2022

1 Code a simple PSO to solve the problem

1.1 Defining problem

Our problem is to minimize objective/fitness function (z) or in other words finding the global minimum for z. Therefore, first I have defined our objective/fitness function which is given as in 1.

$$z = \left(4 - 2.1x^2 + \frac{x^4}{3}\right) x^2 + xy + \left(-4 + 4y^2\right) y^2 \tag{1}$$

I have implemented this equation into MATLAB as in 1.

```
%% We need an objective function to approximate its global optimum
function [z] = ObjectiveFunction(position) % I am going to pass a vector "position"
    x = position(1);
    y = position(2);
    z = (4-2.1*x^2 + x^4/3)*x^2 + x*y + (-4 + 4*y^2)*y^2;
end
```

Algorithm 1: Define fitness/objective function in MATLAB

After, in the main code, I call the objective function and determine the lower and upper bounds for the objective/fitness function. The lower and upper bounds was given in the homework as x lies between ± 3 and y lies between ± 2.

```
%% Problem definition
% Define Objective function
fobj = @ObjectiveFunction;

% Details of the objective function
nVar = 2;
ub = [3 2];
lb = [-3 -2];
```

Algorithm 2: Call fitness/objective function in MATLAB

1.2 Algorithm parameters

The algorithm parameters are:

- **Number of paticles (N):** Number of particles usually between 10 and 50. I have chosen this value as 10.
- **Importance of personal best value (C1):** It has been chosen as 2.
- **Importance of neighborhood best value (C2)**: It has been chosen as 2 since *c1 + c2* is empirically chosen as 4.

- **Position of agent a_i in solution space (p_i)**
- **Velocity of agents a_i (v_i)**
- **Maximum number of iteration (maxIter):** This value has been found experimentally.
- **Initial weights**: Upper bound for weight (wMax) has been chosen as 0.9 and lower bound for weight (wMin) has been chosen as 0.2.
- **Velocity limit**: As a stopping criterion, the velocity is limited. The maximum velocity limit is chosen as $(ub - lb) * 0.2$. Also, the minimum velocity limit is chosen as $-Vmax$.

1.3 Initialize a population

I have created population of agents (particles) and initialized them as in algorithm 3. You can see that the swarm has only one global optimum. However, the personal best values are updated for every iteration.

```
% Initialize the swarm
for i = 1:N
    Swarm.Particle(i).Position = (ub-lb).*rand(1,nVar)+lb;
    Swarm.Particle(i).Velocity = zeros(1,nVar);

    % Personal Best of particle
    Swarm.Particle(i).pBest.Position = zeros(1,nVar);
    Swarm.Particle(i).pBest.Objective = inf; % Because after we will found a small value

    % Global Best of swarm
    Swarm.gBest.Position = zeros(1,nVar);
    Swarm.gBest.Objective = inf;

end
```

Algorithm 3: Initialize a population

1.4 Velocity update equation

Each particle's position is updated according to the objective function. For each iteration if current objective function is smaller than global best or personal best, we have to update the current position and objective function. In order to calculate position vectors, we need to calculate velocity. In order to update velocity, we first need to update the weights as in equation 2.By using these weights, we update the velocities using formula 3.Then, we obtain the updated position vectors.

$$w = wMax - t((wMax - wMin)/maxIter) \tag{2}$$

$$\mathbf{v}_i^{t+1} = \underbrace{\mathbf{w}v_i^t}_{\text{inertia}} + \underbrace{\mathbf{c}_1\mathbf{rand}\left(\text{pBest}_i^t - \mathbf{p}_i^t\right)}_{\text{personal influence}} + \underbrace{\mathbf{c}_2\mathbf{rand}\left(\text{gBest}^t - \mathbf{p}_i^t\right)}_{\text{social influence}} \tag{3}$$

1.5 Stopping criterion

The stopping criterion is done by limiting the velocity. If the particle's current velocity exceeds the velocity limits (which are Vmax and Vmin) we inhibit the loop going further away. The same thing has been done for position, meaning that we do not want the position to exceed the position limits (ub and lb).

1.6 Results

From figure 1, it can be seen that the objective function converges to global minima which is -1.0316 for the given objective function. The results can be seen more clearly, by running the code in appendix 1.6.

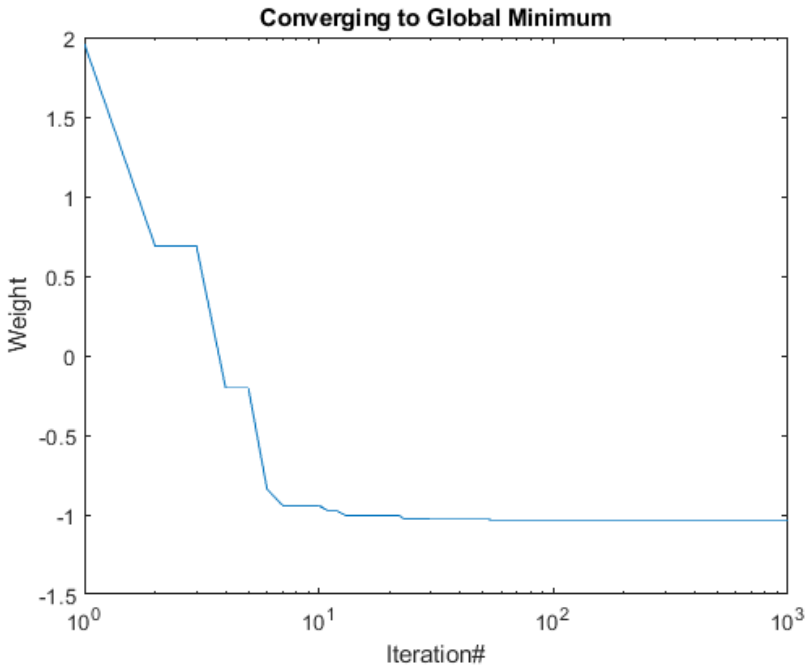

Figure 1: Converging to global minimum.

Appendix

Fitness/Objective Function

```
%% We need an objective function to approximate its global optimum
function [z] = ObjectiveFunction(position) % I am going to pass a
    vector "position"
     x = position(1);
     y = position(2);
     z = (4-2.1*x^2 + x^4/3)*x^2 + x*y + (-4 + 4*y^2)*y^2;
end
```

Simulation code

```
clc;clear all;close all;
%% Problem definition
% Define Objective function
fobj = @ObjectiveFunction;

% Details of the objective function
nVar = 2;
ub = [3 2];
lb = [-3 -2];

%% PSO Algorithm parameters
% Number of particles -- usually between 10 and 50
N = 10;

% c1 + c2 = 4 (empirically chosen value)
% importance of personal best value
c1 = 2;
% importance of neighborhood best value
c2 = 2;

% Maximum number of iteration
maxIter = 1000; % Find by experiment

% Initial weights
wMax = 0.9; % upper bound for weight
wMin = 0.2; % lower bound for weight

%% Limit velocity

Vmax = (ub-lb).*0.2;
Vmin = -Vmax;

%% The PSO
% Initialize the swarm
for i = 1:N
     Swarm.Particle(i).Position = (ub-lb).*rand(1,nVar)+lb;
     Swarm.Particle(i).Velocity = zeros(1,nVar);

     % Personal Best of particle
     Swarm.Particle(i).pBest.Position = zeros(1,nVar);
```

```
41      Swarm.Particle(i).pBest.Objective = inf; % Because after we
    will found a small value(smaller than inf)
42
43      % Global Best of swarm
44      Swarm.gBest.Position = zeros(1,nVar);
45      Swarm.gBest.Objective = inf;
46
47  end
48
49  % PSO Loop
50  for t = 1:maxIter
51
52      for i = 1:N
53          currentPosition = Swarm.Particle(i).Position;
54          Swarm.Particle(i).Objective = fobj(currentPosition);
55          currentObjective = Swarm.Particle(i).Objective;
56
57          % UPDATE PERSONAL BEST if current objective is smaller than
     its value
58          % Objective function comparison for personal best
59          if Swarm.Particle(i).Objective < Swarm.Particle(i).pBest.
    Objective
60              Swarm.Particle(i).pBest.Position = currentPosition;
61              Swarm.Particle(i).pBest.Objective = currentObjective;
62          end
63
64          % UPDATE GLOBAL BEST if current objective is smaller than
    its value
65          % Objective function comparison for global best
66          if Swarm.Particle(i).Objective < Swarm.gBest.Objective
67              Swarm.gBest.Position = currentPosition;
68              Swarm.gBest.Objective = currentObjective;
69          end
70      end
71
72      % Particle Update Rule --> Update position and velocity vectors
73      % In order to calclate position vectors, we need to calculate
    velocity
74      % In order to calculate velocity, we update initial weight
75      w = wMax - t.*((wMax-wMin)/maxIter);
76      for i = 1:N
77          Inertia_Part = w.*Swarm.Particle(i).Velocity;
78          PersonalInf_Part = c1.*rand(1,nVar).*(Swarm.Particle(i).
    pBest.Position - Swarm.Particle(i).Position);
79          SocailInf_Part = c2.*rand(1,nVar).*(Swarm.gBest.Position -
    Swarm.Particle(i).Position);
80          Swarm.Particle(i).Velocity = Inertia_Part +
    PersonalInf_Part + SocailInf_Part;
81
82          % Consider Velocity Limitation -->Must not go out of bound!
83          % Check velocities
84          ind_ExceedVmax = find(Swarm.Particle(i).Velocity > Vmax);
85          ind_ExceedVmin = find(Swarm.Particle(i).Velocity < Vmin);
86
87          Swarm.Particle(i).Velocity(ind_ExceedVmax) = Vmax(
    ind_ExceedVmax);
```

```
88          Swarm.Particle(i).Velocity(ind_ExceedVmin) = Vmin(
    ind_ExceedVmin);
89
90          % The updated particle position
91          Swarm.Particle(i).Position = Swarm.Particle(i).Position +
    Swarm.Particle(i).Velocity;
92
93          % Check positions -->Must not go out of bound!
94          ind_Exceedub = find(Swarm.Particle(i).Position > ub);
95          ind_Exceedlb = find(Swarm.Particle(i).Position < lb);
96
97          Swarm.Particle(i).Position(ind_Exceedub) = ub(ind_Exceedub)
    ;
98          Swarm.Particle(i).Position(ind_Exceedlb) = lb(ind_Exceedlb)
    ;
99
100      end
101
102      msg = ['Iteration# ', num2str(t) , ' Swarm.gBest.Objective = '
    , num2str(Swarm.gBest.Objective)];
103      disp(msg);
104
105      cgCurve(t) = Swarm.gBest.Objective;
106  end
107
108
109  semilogx(cgCurve);
110  title('Converging to Global Minimum')
111  xlabel('Iteration#')
112  ylabel('Weight')
```