Article

Particle swarm optimization: A Matlab algorithm

WenJun Zhang
School of Life Sciences, Sun Yat-sen University, Guangzhou 510275, China; International Academy of Ecology and Environmental Sciences, Hong Kong
E-mail: zhwj@mail.sysu.edu.cn, wjzhang@iaees.org

Received 9 May 2021; *Accepted 12 October 2021; Published online 20 August 2022*; *Published 1 December 2022*

Abstract
In present study, the Matlab algorithm and full codes for particle swarm optimization was given. An example was demonstrated.

Keywords particle swarm optimization; Matlab algorithm; software.

Selforganizology
ISSN 2410-0080
URL: http://www.iaees.org/publications/journals/selforganizology/online-version.asp
RSS: http://www.iaees.org/publications/journals/selforganizology/rss.xml
E-mail: selforganizology@iaees.org
Editor-in-Chief: WenJun Zhang
Publisher: International Academy of Ecology and Environmental Sciences

1 Introduction

As early as in 1975, Wilson proposed the swarm theory (Wilson, 1975). In a swarm, each individual may share the other individuals' discovery and experience to escape from predators and obtain food. The theory formed the theoretical foundation of particle swarm optimization that widely used around the world (Angeline, 1998; Kennedy, 2000; Zhang et al., 2006; Eberhart and Shi, 2007; Lin et al., 2008; Imran et al., 2013; Zhang, 2016). Particle swarm optimization is a kind of methods in selforganizology (Zhang, 2013a, b, 2015, 2016).

In a bird swarm, each individual can locate itself in the swarm. Every individual will perceive the flight movements of neighbor individuals to update its own flight trajectory, which makes the whole swarm appear to be controlled by a central system. Thus we may consider that every bird must perceive three aspects: its own location, the locations of two or three neighbors, and the flight trajectory of whole swarm (Zhang, 2016). Based on this, Reynolds (1987) developed a distributed behavioral model. Kennedy and Eberhart (1995) proposed Particle Swarm Optimization to deal with optimization problems of continuous functions (Yang and Li, 2004). Particle swarm optimization searches the optimal solution by cooperation and competition between particles (i.e., individuals). Every particle is treated as a point in a higher dimensional space. Each particle moves at a certain velocity in the solution space, and towards to its historically best location and the historically best location of neighbors, in order to evolve as a candidate solution (Zhang, 2016).

In present study, I presented the Matlab algorithm and full codes for particle swarm optimization. An example was demonstrated for better use.

2 Methods

2.1 Particle Swarm Optimization

The particle swarm optimization includes the following procedures and contents:

2.1.1 Determine the objective function

Determine the objective function $f(x)$ that searches for the maximum value in the given intervals, e.g., [a,b]. For example, in this study, the objective function to search for the optimal value is

$$f(x)=x*sin(x)*cos(x)$$

The goal is to find the maximum value of the function on the interval [0,12].

2.1.2 Set parameters

The main parameters of the particle swarm optimization include:

(1) Space dimension: the space dimension of particle swarm search, which is the number of independent variables of the objective function.

(2) Particle swarm size: The larger the initial particle swarm, the better the convergence process. However, if the initial particle swarm is too large, it will affect the particle velocity. The initial particle swarm size can be 50~1000.

(3) Location constraint: constrain the space of particle swarm search, that is, the ranges of the independent variables, e.g., [a,b].

(4) Velocity constraint: If the particle velocity is too fast, it may directly cross the location of optimal solution. If the particle velocity is too slow, it will slow down the convergence process. Therefore, it is necessary to set a reasonable velocity constraint.

(5) Number of iterations: If the number of iterations is too small, the solution will be unstable, however too many iterations will waste time. The number of iterations is generally 100~5000. For complex problems, iterations can be increased accordingly.

(6) Inertia weight: The inertia weight reflects the influence of the particle's historical performance on the current, it is generally 0.5~1.

(7) Learning factor: The learning factor should be determined according to the ranges of the independent variables. There are two types of learning factors: particle and particle swarm learning factors. Generally, the value of 0~5 can be taken.

2.1.3 Initialize particle swarm

The location is restricted to, e.g., [a,b]. Since the problem in this study is relatively simple, the particle swarm size s=50, and the number of iterations is 100. In this example, the space dimension d=1. The initialization of location and velocity is to randomly generate the $s\times d$ data matrix within the location and velocity constraints. In this example, the location initialization is to randomly generate a 50×1 data matrix within [a,b]. Velocity initialization is to randomly generate a 50×1 data matrix within [-1,1]. The Velocity constraint is to ensure that the particle's step size does not exceed the constraint, which can be valued at [-1,1].

A feature of particle swarm is that it records the historical optimum of single particles and the historical optimum of the particle swarm. Therefore, the optimal location and optimal value corresponding to the two also need to be initialized. Among them, the historical optimal location of single particles can be initialized as the current location, and the historical optimal position of particle swarm can be initialized as the origin. For the optimal value, it is initialized to negative infinity if the maximum value is sought, and positive infinity if the minimum value is sought.

For each search, the current and optimal solutions need to be compared with historical records. If the historical optimal value is exceeded, the historical optimal locations and optimal solutions of single particles and particle swarm are updated.

2.1.4 Update velocity and location

Updating velocity and location is the core of particle swarm optimization. The update rules are as follows:

$v_{id}=w\cdot v_{id}+c_1\cdot r_1\cdot(p_{id}-x_{id})+c_2\cdot r_2\cdot(p_{gd}-x_{id})$

$x_{id}=x_{id}+v_{id}$

where, w: inertia weight; c_1: single particle's learning factor; c_2: particle swarm's learning factor; r_1, r_2: random values in [0,1].

2.2 Matlab Algorithm

The Matlab codes for the particle swarm optimization above are as follows (also find supplementary material):

```
clc
clear;
close all;
%%Initialization process
d=input('Input dimension of search space (e.g., the number of independent variables in the function): ');
s=input('Input the initial size of particle swarm (e.g., 50, 80, 1000, etc): ');
pliml=input('Input the lower constraint for particles location (e.g., the lower limit of independent variables for searching maximal function value): ');
plimu=input('Input the upper constraint for particles location (e.g., the upper limit of independent variables for searching maximal function value): ');
miter=input('Input the maximum iterations (e.g., 100, 200, 5000, etc): ');
inw=input('Input the inertia weight (e.g., 0.5, 0.6, 1, etc): ');
c1=input('Input the learning factor of single particles (e.g., 0.1, 1, 3, 5, etc): ');
c2=input('Input the learning factor of particle swarm (e.g., 0.1, 0.5, 1, 3, 5, etc): ');
plim=[pliml plimu];
vlim=[-1,1];          %Velocity constraint for single particles
x=plim(1:d,1)+(plim(1:d,2)-plim(1:d,1)).*rand(s,d);    %Location of initial particle swarm
v=rand(s,d);          %Velocity of initial particle swarm
xm=x;                 %Historical optimum location of single particles
ym=zeros(1,d);     %Historical optimum location of particle swarm
fxm=zeros(s,1);     %Historical optimum function value of single particles
fym=-inf;             %Historical optimum function value of particle swarm
figure(1)
plot(xm,f(xm),'k.');
title('Initial Locations of Particles');
xlabel('x');
ylabel('f(x)');
%%Update process
iter=1;
record=zeros(miter,1);       %Recorder
while (iter<=miter)
fx=f(x);                      %Current function value (i.e., fitness)
for i=1:s
if (fxm(i)<fx(i))
fxm(i)=fx(i);                 %Update historial optimum function value of single particles
xm(i,:)=x(i,:);                   %Update historial optimum location of single particles
```

```
end
end
if (fym<max(fxm))
[fym,nmax]=max(fxm);          %Update historial optimum function value of particle swarm
ym=xm(nmax,:);                %Update historial optimum location of particle swarm
end
v=v*inw+c1*rand*(xm-x)+c2*rand*(repmat(ym,s,1)-x);      %Update velocity
%Velocity for boundary
v(v>vlim(2))=vlim(2);
v(v<vlim(1))=vlim(1);
x=x+v;                        %Update location
%Location for boundary
x(x>plim(2))=plim(2);
x(x<plim(1))=plim(1);
record(iter)=fym;             %Record maximal function value
x0=pliml:0.01:plimu;
figure(2)
plot(x0,f(x0),'b-',x,f(x),'k.');
title('Changes of Location')
xlabel('x');
ylabel('f(x)');
pause(0.1)
iter=iter+1;
end
figure(3)
plot(record);
title('Convergence Process')
x0=pliml:0.01:plimu;
figure(4)
plot(x0,f(x0),'b-',x,f(x),'k.');
title('Final Location')
xlabel('x');
ylabel('f(x)');
disp(['Maximal function value f(x)=',num2str(fym)]);
disp(['x=',num2str(ym)]);

function fx=f(x)
fx=x.*sin(x).*cos(x);         %The function to find maximal function value
```

3 Example Demonstration

Using the function and location constraint above, the following parameters of particle swarm optimization are set in Matlab program:

```
Input dimension of search space (e.g., the number of independent variables in the function): 1
Input the initial size of particle swarm (e.g., 50, 80, 1000, etc): 50
```

Input the lower constraint for particles location (e.g., the lower limit of independent variables for searching maximal function value): 0

Input the upper constraint for particles location (e.g., the upper limit of independent variables for searching maximal function value): 12

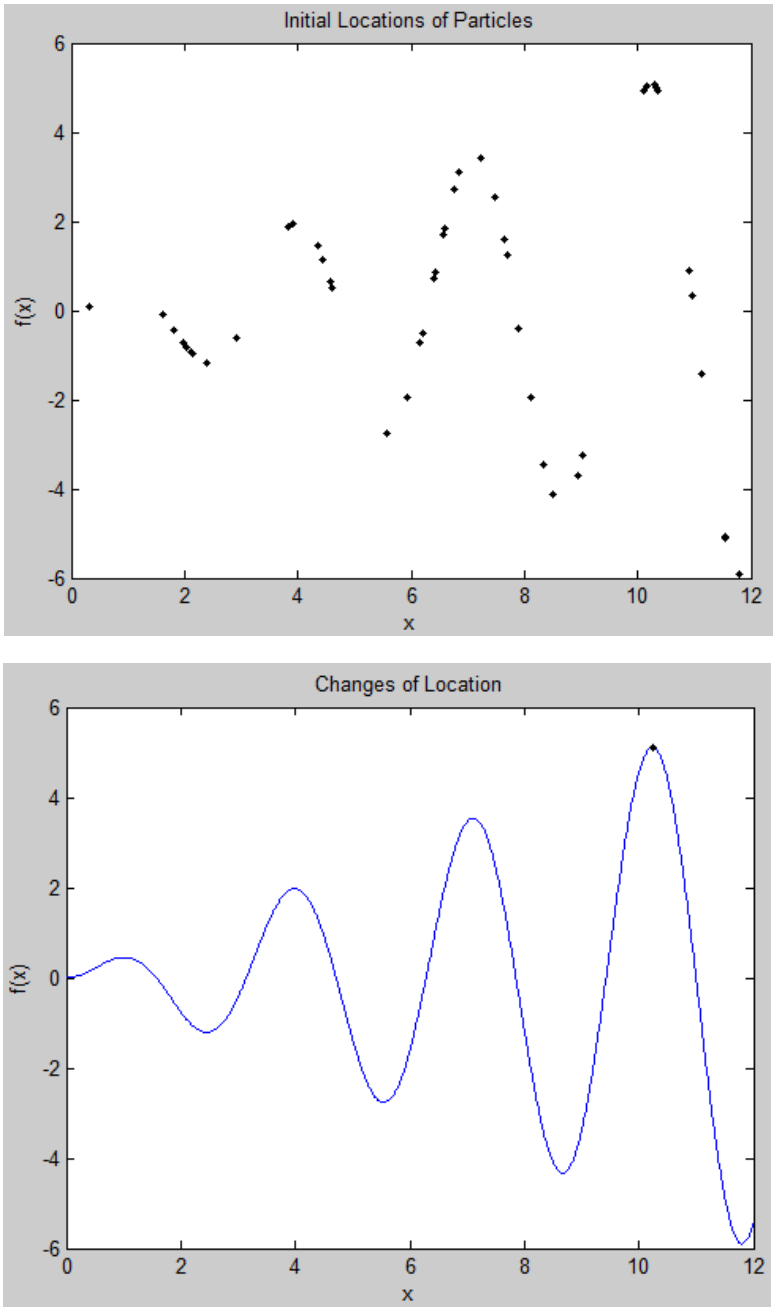
Input the maximum iterations (e.g., 100, 200, 5000, etc): 100

Input the inertia weight (e.g., 0.5, 0.6, 1, etc): 0.5

Input the learning factor of single particles (e.g., 0.1, 1, 3, 5, etc): 0.5

Input the learning factor of particle swarm (e.g., 0.1, 0.5, 1, 3, 5, etc): 0.5

The results showed that the optimum value (maximal function value) is $f(x_{max})$=5.1112, and x_{max} =10.2346, which coincide with true values (Fig. 1).



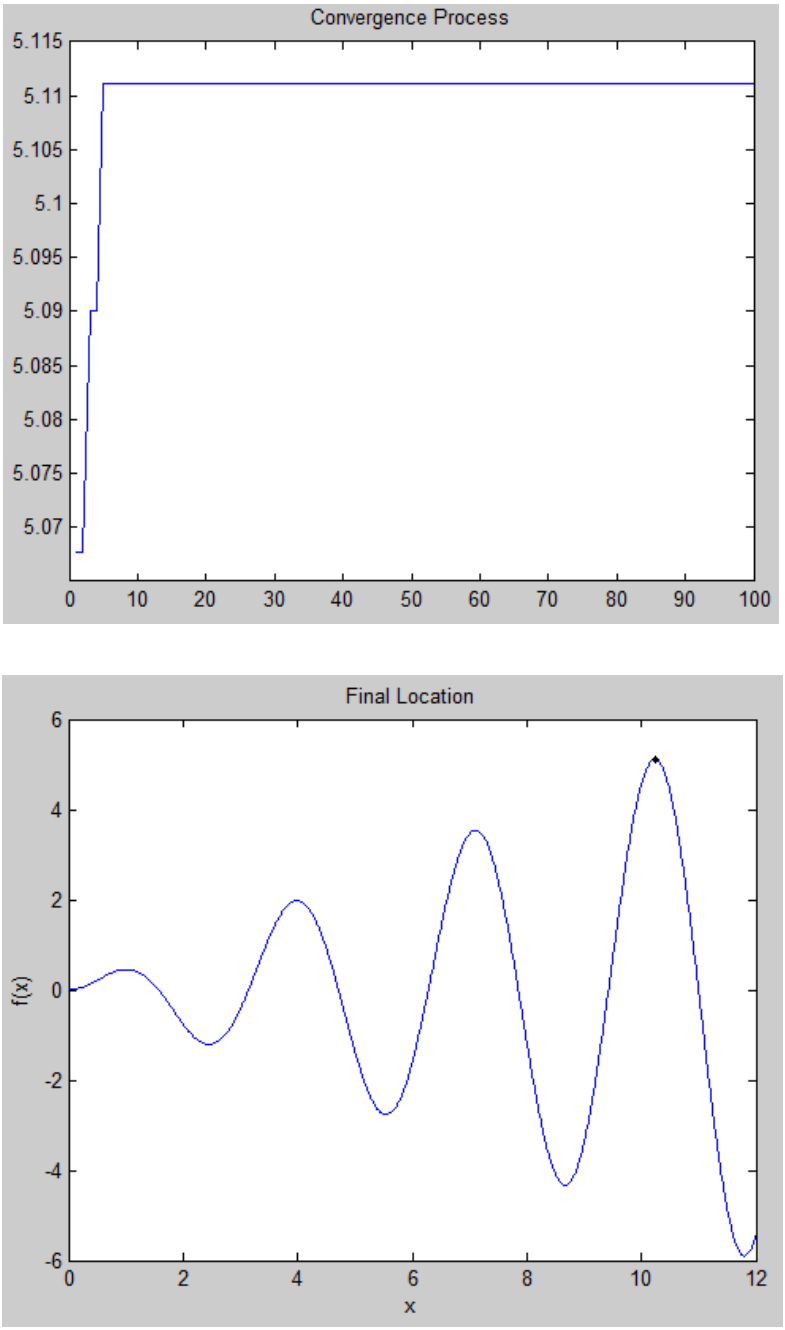


Fig. 1 Results of particle swarm optimization.

4 Discussion

The present Matlab algorithm is actually for 1D problems. The algorithm can be easily improved to solve *n*-dimension problems.

Acknowledgment

We are thankful to the support of The National Key Research and Development Program of China (2017YFD0201204), and Discovery and Crucial Node Analysis of Important Biological and Social Networks (2015.6-2020.6), from Yangling Institute of Modern Agricultural Standardization.

References

Angeline PJ. 1998. Using Selection to Improve Particle Swarm Optimization. In: Proceedings of IEEE International Conference on Neural Networks, 84-89

Eberhart RC, Shi YH. 2007. Computational Intelligence: Concepts to Implementations. Morgan Kaufmann, MA, USA

Imran M, Hashim R, Abd Khalid NE. 2013. An overview of particle swarm optimization variants. Procedia Engineer, 53: 491-496

Kennedy J. 2000. Stereotyping: Improving particle swarm performance with cluster analysis. In: Proceedings of the 2000 Congress on Evolutionary Computation, 1 and 2: 1507-1512

Kennedy J, Eberhart, R. 1995. Particle Swarm Optimization. In : Proceedings of IEEE International Conference on Neural Networks IV. 1942-1948, Perth, Australia

Ling SH, Iu HHC, Leung FHF, et al. 2008. Improved hybrid particle swarm optimized wavelet neural network for modeling the development of fluid dispensing for electronic packaging. IEEE Transactions on Industrial Electronics, 5: 3447-3460

Reynolds CW. 1987. Flocks, herds and schools: A distributed behavioral model. Computer Graphics, 21(4): 25-34

Wilson EO. 1975. Sociobiology: The New Synthesis. Belknap Press, Cambridge, MA, USA

Yang W, Li QQ. 2004. Survey on particle swarm optimization algorithm. Engineering Science, 6(5): 87-94

Zhang MF, Shao C, Gan Y, et al. 2006. Hybrid artificial fish swarm optimization algorithm based on mutation operator and simulated annealing. Acta Electronica Sinica, 34(8): 1381-1385

Zhang WJ. 2013a. Self-organization: Theories and Methods. Nova Science Publishers, New York, USA

Zhang WJ. 2013b. Selforganizology: A science that deals with self-organization. Network Biology, 3(1): 1-14

Zhang WJ. 2015. A generalized network evolution model and self-organization theory on community assembly. Selforganizology, 2(3): 55-64

Zhang WJ. 2016. Selforganizology: The Science of Self-Organization. World Scientific, Singapore