Article

Genetic algorithm: A Matlab software

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 13 June 2021; *Accepted 28 August 2022; Published online 1 September 2022*; *Published 1 June 2023*

Abstract
In present study, the Matlab software for a genetic algorithm was given. An example was demonstrated for easy use.

Keywords genetic algorithm; Matlab 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

Genetic algorithms (GAs) are stochastic and global search methods that mimic the metaphor of natural biological evolution (Wang and Cao, 2002; Feng and Zhang, 2004; Eberhart and Shi, 2007; Chipperfield et al., 2014; Zhang, 2013a-b, 2016). They are essentially evolutionary algorithms and are categorized to methods of selforganizology (Zhang, 2016, 2022). GAs are substantially adaptive plans. In an adaptive plan, structures in a search space are selected and continuously modified by some rules according to the quality of their performance in the past (Zhang, 2016). In a GA, a set of structures is a population and each structure is a genotype/individual. Adaptive operators in GAs, i.e., genetic operations, include selection, crossover and mutation. GAs operate on a population using the principle of survival of the fittest, in order to produce better and better approximations to a solution (Feng and Zhang, 2004; Yadav et al., 2017a-c, 2018, 2019). Genetic operations are used in each generation to mimic natural adaptation. It leads to the evolution of individuals that are better suited to their environment than their parent individuals (Chipperfield et al., 2014; Zhang, 2016).

GAs differ from conventional search and optimization methods. The most distinct differences between GAs and conventional methods include (Chipperfield et al., 2014; Zhang, 2016): (1) GAs search a population in parallel; (2) GAs do not require derivative information or other auxiliary knowledge, only the objective function and corresponding fitness values affect the directions of search; (3) GAs use probabilistic transition rules rather than deterministic ones, and (4) GAs work on an encoding of the parameter set, rather than the parameter set itself.

In this study, the Matlab software for a genetic algorithm was given. An example was demonstrated for easy use.

2 Methods

2.1 Genetic algorithm

The procedures of the genetic algorithm (Zhang, 2016) used in this study are:

(1) Initialize the population. A population is a set of multiple individuals.

(2) Set up a crossover pool, and put some of the most adaptable individuals (i.e., the individuals with maximum fitness) in the population into the crossover pool.

(3) Update and record the optimal individuals of the current generation in the crossover pool.

(4) Selection. The individuals to be crossed are selected from the crossover pool. Individuals whose crossover will occur are randomly selected according to their fitness. Individuals with higher fitness have a greater chance of being selected, but individuals with relatively low fitness also have a chance of being selected. Use roulette selection method to make selection. In the roulette selection method, a roulette wheel is generated according to the individuals' selection probabilities, wherein the size of each area of the roulette wheel is proportional to the individual's selection probability. Then a random number is generated, and find which area of the roulette wheel it falls into, and the corresponding individual in that area is selected for crossover. Obviously, the individual with the higher selection probability are more likely to be selected and have a greater chance of obtaining crossover. From this, the proportion of each individual in the crossover pool is calculated. Subsequently, two individuals are drawn using random numbers.

(5) Crossover. The two selected individuals are crossed. Here, the two-location intersection method is used:

(a) Select two different locations arbitrarily, and exchange individual segments between the two locations.

(b) Record conflict location subscripts before crossing.

(c) Conflict locations are resolved after crossing.

(6) Mutation. Set a small probability and two new individuals mutate at the probability. Here, the mutation is set to be simple segment inversion.

(7) Calculate the fitness and put the new individual with better fitness into the crossover pool. Iterate in this way until the crossover pool reaches the maximum number of individuals in the population per generation.

2.2 Matlab software

The Matlab software of genetic algorithm here is for Traveling Salesman Problem (TSP). Suppose there is a traveling salesman who wants to visit *n* nodes. He must choose the path he wants to take. The limit of the path is that each node can only be visited once, and he must return to the original node. The path selection goal is to calculate the path distance to be the minimum value among all paths. The Matlab software codes for the genetic algorithm above are as follows (also find supplementary material):

```
clear all;
clc;
Cross_Size=input('Input size of crossover pool (e.g., the number of individuals to be crossed. e.g., 30, etc.): ');
iterMax=input('Input the number of maximum iterations (e.g., 500, 2000, 5000, etc.): ');
datafile=input('Input the data file name (In the data file the first column is node IDs, the second and third columns are for x and y coordinates of the nodes): ','s');
datafull=load(datafile);
data=datafull(:,2:3);
Node_Number=size(data,1);
pool=zeros(Cross_Size,Node_Number);
CrossPool=[];
for i=1:Node_Number
AdjMat(i,i)=0;
```

```
for j=1:Node_Number
if (i~=j)
AdjMat(i,j)=sqrt(sum((data(i,:)-data(j,:)).^2));
end
end
end
for i=1:Cross_Size
pool(i,:)=randperm(Node_Number);
end
Path_Best=pool(1,:);
len=zeros(Cross_Size,1);
fitness=zeros(Cross_Size,1);
num=0;
while (num<iterMax)
for i=1:Cross_Size
len(i,1)=AdjMat(pool(i,Node_Number),pool(i,1));
for j=1:(Node_Number-1)
len(i,1)=len(i,1)+AdjMat(pool(i,j),pool(i,j+1));
end
end
maxlen=max(len);
minlen=min(len);
path=find(len==minlen);
Path_Best=pool(path(1,1),:);
for i=1:length(len)
fitness(i,1)=(1-((len(i,1)-minlen)/(maxlen-minlen+0.001)));
end
count=0;
for i=1:Cross_Size
if (fitness(i,1)>=rand)
count=count+1;
CrossPool(count,:)=pool(i,:);
end
end
[Indiv_Count,l]=size(CrossPool);
while (Indiv_Count<Cross_Size)
Random_Num=randperm(count);
a=CrossPool(Random_Num(1),:);
b=CrossPool(Random_Num(2),:);
w=ceil(Node_Number/10);
p=unidrnd(Node_Number-w+1);
for i=1:w
x=find(a==b(p+i-1));
y=find(b==a(p+i-1));
temp=a(p+i-1);
```

```
a(p+i-1)=b(p+i-1);
b(p+i-1)=temp;
temp=a(x);
a(x)=b(y);
b(y)=temp;
end
p1=floor(1+Node_Number*rand());
p2=floor(1+Node_Number*rand());
while (p1==p2)
p1=floor(1+Node_Number*rand());
p2=floor(1+Node_Number*rand());
end
tmp=a(p1);
a(p1)=a(p2);
a(p2)=tmp;
tmp=b(p1);
b(p1)=b(p2);
b(p2)=tmp;
CrossPool=[CrossPool;a;b];
[Indiv_Count,l]=size(CrossPool);
end
if (Indiv_Count>Cross_Size)
CrossPool=CrossPool(1:Cross_Size,:);
end
pool=CrossPool;
pool(1,:)=Path_Best;
clear CrossPool;
items=[Path_Best,Path_Best(1)];
scatter(data(Path_Best,1),data(Path_Best,2))
for i=1:length(Path_Best)
text(data(Path_Best(i),1),data(Path_Best(i),2),num2str(items(i)))
end
hold on
plot(data(items,1),data(items,2))
text(1500,1100,['Distance=',num2str(minlen)]);
num=num+1;
P_length(num)=minlen;
end
figure
plot(P_length)
disp('The best path is:');disp([Path_Best,Path_Best(1)])
disp(['The smallest distance:',num2str(minlen)]);
```

3 Example Demonstration

Using the algorithm and software above, the following parameters of genetic algorithm are set in Matlab software:

Input size of crossover pool (e.g., the number of individuals to be crossed. e.g., 30, etc.): 30

Input the number of maximum iterations (e.g., 2000, 5000, etc.): 100

Input the data file name (In the data file the first column is node IDs, the second and third columns are for x and y coordinates of the nodes): data.txt

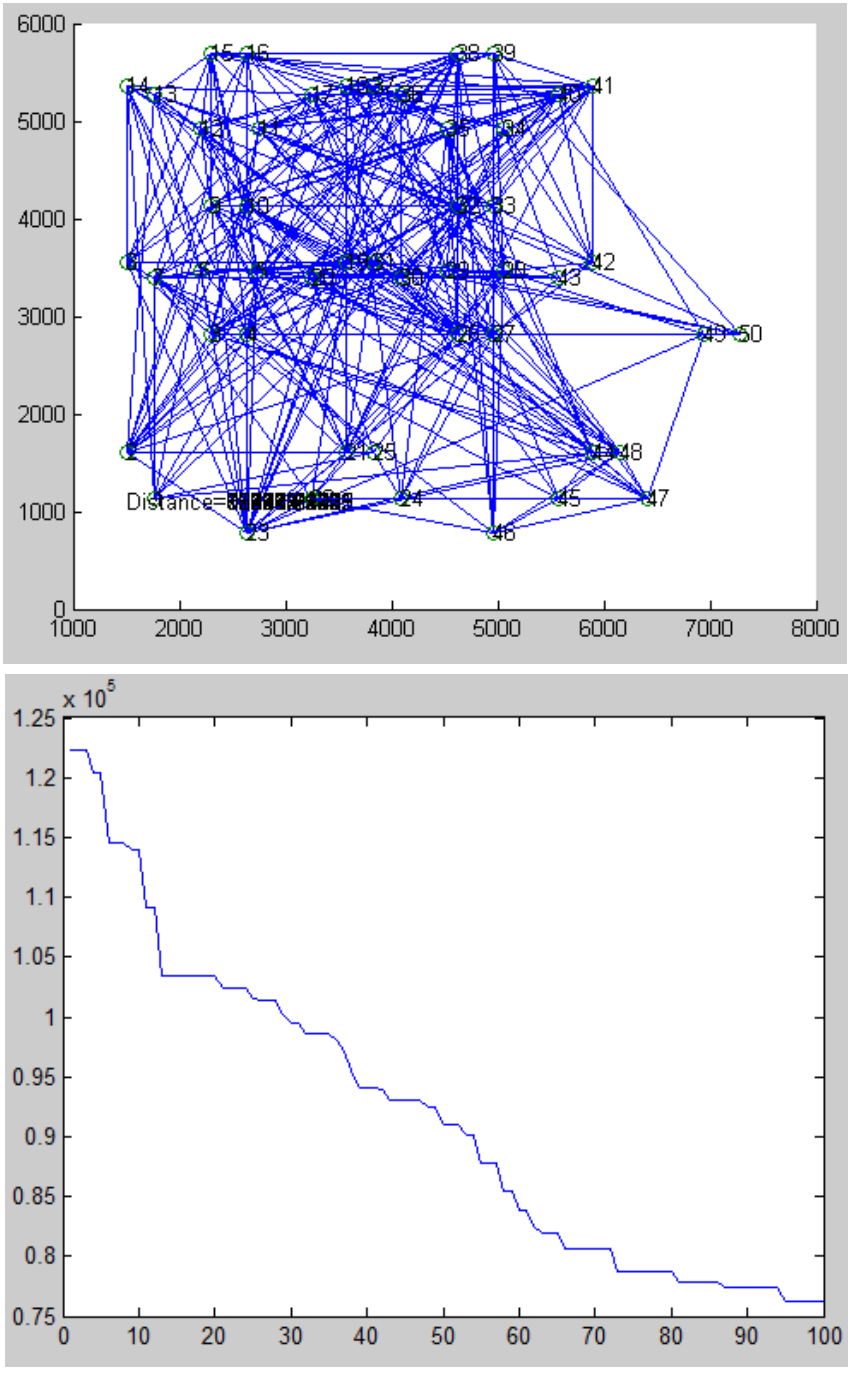

Fig. 1 Results of genetic algorithm.

The best path is (data are supplied in supplementary material):

37->32->40->33->27->2->25->1->19->30->48->44->3->10->11->16->15->13->12->20->7->9->8->14->6->26->41->35->36->17->39->38->5->4->23->21->31->43->42->34->29->50->49->47->45->46->22->24->28->18->37.

The smallest distance is calculated as 76201.0433. The iteration effects are indicated in Fig. 1.

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

Chipperfield A, Fleming P, Pohlheim H, et al. 2014. Genetic Algorithm Toolbox User's Guide. MathWorks, USA

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

Feng YJ, Zhang WJ. 2004. Network software of Genetic BP algorithm and its application in biodiversity research. Application Research of Computers, Suppl: 387-389

Wang XP, Cao LM. 2002. Genetic Algorithms: Theory, Applications and Software Implementation. Xian Jiaotong University Press, Xian, China

Yadav AS, Maheshwari P, Swami A, Garg A. 2017a. Analysis of six stages supply chain management in inventory optimization for warehouse with artificial bee colony algorithm using genetic algorithm. Selforganizology, 4(3): 41-51

Yadav AS, Maheshwari P, Swami A, Pandey G. 2018. A supply chain management of chemical industry for deteriorating items with warehouse using genetic algorithm. Selforganizology, 5(1-2): 41-51

Yadav AS, Swami A, Kher G. 2019. Blood bank supply chain inventory model for blood collection sites and hospital using genetic algorithm. Selforganizology, 6(3-4): 13-23

Yadav AS, Swami A, Gupta CB, Garg A, 2017b. Analysis of electronic component inventory optimization in six stages supply chain management for warehouse with ABC using genetic algorithm and PSO. Selforganizology, 4(4): 52-64

Yadav AS, Swami A, Kher G, Garg A, 2017c. Analysis of seven stages supply chain management in electronic component inventory optimization for warehouse with economic load dispatch using genetic algorithm. Selforganizology, 4(2): 18-29

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

Zhang WJ. 2022. Particle swarm optimization: A Matlab algorithm. Selforganizology, 9(3-4): 35-41