REPORT

LABORATORY OF EVOLUTIONARY ALGORITHMS

Genetic Algorithm for Traveling Salesman Problem

Authors:
Angelika Ucherek
Piotr Pawełko
MAKRO – Data Science

Date of the exercise: 14.03.2022, 28.03.2022 r.
Date of the report submission: 04.04.2022 r.

1. Source code

1.1. Main source code:

```
%% GA main
clear all;

% Parameters:
 P=100% 250; %100, 300, 500 % number of chromosomes
 n=0.5% 0.8; % 0.5, 0.7, 0.9   percentage of the population that will
become parents
 pm=0.1% 0.2; % 0.1, 0.7, 0.9  mutations rate
 Tmax=1000; % number of iterations
 N=10; % number of cities

% Initial population:
P0=[1 2 3 4 5 6 7 8 9 10];
Population=initPop(P0,P);
% Distance matrix for map number 1:
x = [0 3 6 7 15 12 14 9 7 0];
y = [1 4 5 3 0 4 10 6 9 10];
plot(x,y,'black*','MarkerSize',4,'LineWidth',3);
title('Map of cities (number 1)'); hold on;
xlabel('Coord x');
ylabel('Coord y');
xlim([min(x)-2,max(x)+2]);
ylim([min(y)-2,max(y)+2]);
labels = string(1:10);
text(x-.1,y-.3, labels);
% adding connections between different points:
connect=nchoosek(1:length(x),2)';
plot(x(connect),y(connect),"Color",'#FFC0CB');hold on;

% compute the distance matrix [10x10]:
Distance=zeros(length(P0)); %initialize distance matrix
for i=1:N
    for j=1:N
        Distance(i,j) = sqrt((x(i)-x(j))^2+(y(i)-y(j))^2);
    end
end

% main itertion:
for i=1:Tmax

    % Evaluation/ cost value:
    [fi,di]=evaluation(Population, Distance);
    mf=max(fi);
    ti=(mf-fi);
    ts=sum(ti);

    % select of parent population: n*P
    PPop=n*P ;% Quantity of parents chromosomes

    for i=1:PPop
      roulette=0;
      random=rand()*ts;
        dd=randperm(P);

% first individual who exceed the random value is taken to the
parents
```

```
for k=1:P
         roulette=roulette+ti(dd(k));
         if roulette>=random
          Parents(i,:)=Population(dd(k),:);
         end
      end
    end
    half=length(Parents)/2;
    P1=Parents(1:half,:);
    P2=Parents(half+1:length(Parents),:);

    %Cycle crossover
    for i=1:length(P1)
        Offspring1(i,:)=cycle_cross(P1(i,:),P2(i,:));
        Offspring2(i,:)=cycle_cross(P2(i,:),P1(i,:));
    end
    Offspring=[Offspring1;Offspring2];

    % Mutation
    for j=1:length(Offspring)
      random_pm= rand();
      if random_pm<pm
         pos_mut=randperm(N,2); % random position of mutations
         temp=Offspring(j,pos_mut(1));
         Offspring(j,pos_mut(1))=Offspring(j,pos_mut(2));
         Offspring(j,pos_mut(2))=temp;
      end
    end
    % Evaluation of Offspring
    [fi_O,di_O]=evaluation(Offspring, Distance);
    %  %Join offspring population with original population
    Mixed=[Population;Offspring];

    %Create new population (250 chromosomes) which have min. fi from
mixed population
    Mixed_fi=[fi fi_O];
    [values, index]=mink(Mixed_fi,P);
    New_population=Mixed(index,:);
    fi_new=Mixed_fi(index);
    Population=New_population;
End

 % Show the best distance and sequence:
[value1, index1]=min(fi_new);
Best_way=Population(index1,:)
Minimal_fi=value1
plot(x([Best_way,Best_way(1)]),y([Best_way,Best_way(1)]),"k.",
'MarkerSize',2,'LineWidth',1)
```

1.2. Initial population:

```
function Population = initPop(P0,P)
PP=perms(P0);
[x2 , ~]=size(PP);
Pop=[];
for i=1:P
    Pop1= PP(randi(x2),:);
    Pop=[Pop;Pop1];
end
Population=Pop;
```

1.3. Evaluation

```
function [fi,di]=evaluation(Pi,D)
% Pi = population of P chromosome
% D= matrix of distance beetween every cities
[xP, yP]=size(Pi);
di=zeros(xP,yP);
for j=1:xP
    tour=0;
    %Distance from the last city till first city:
    di(j,1)=D(Pi(j,1),Pi(j,yP));
    for k=1:(yP-1)
      di(j,k+1)=D(Pi(j,k),Pi(j,k+1)); % distance between every point
in given sequence
      tour=tour+di(j,k+1);
    end
f(j)=di(j,1)+tour; % Total distance in a given sequence
end
fi=f;
di=di;
```

1.4. Cycle crossover

```
function O=cycle_cross(P1,P2)
    O=[];
    [~, y1]=size(P1);
    O1 =zeros(1,y1);
    O1(1)=P1(1); pos1=1;

for i=1:length(P1)
     pos1 = find(P1==P2(pos1));
     O1(pos1) = P1(pos1);
     if pos1==1
       empty=find(O1==0);
       O1(empty)=P2(empty);
     end

 end
 O=[O;O1];

OO=O;
```

2. Results

2.1. Task 1:

Parameters:

$P = 250$ → number of chromosomes
$n = 0.8$ → percentage of the population that will become parents
$p_m = 0.2$ → mutations rate
$T_{max} = 1000$ → maximum number of generations

Results:

```
Best_way =
     7     9    10     1     2     3     4     8     6     5

Minimal_fi (minimum distance)=
   55.0441
```

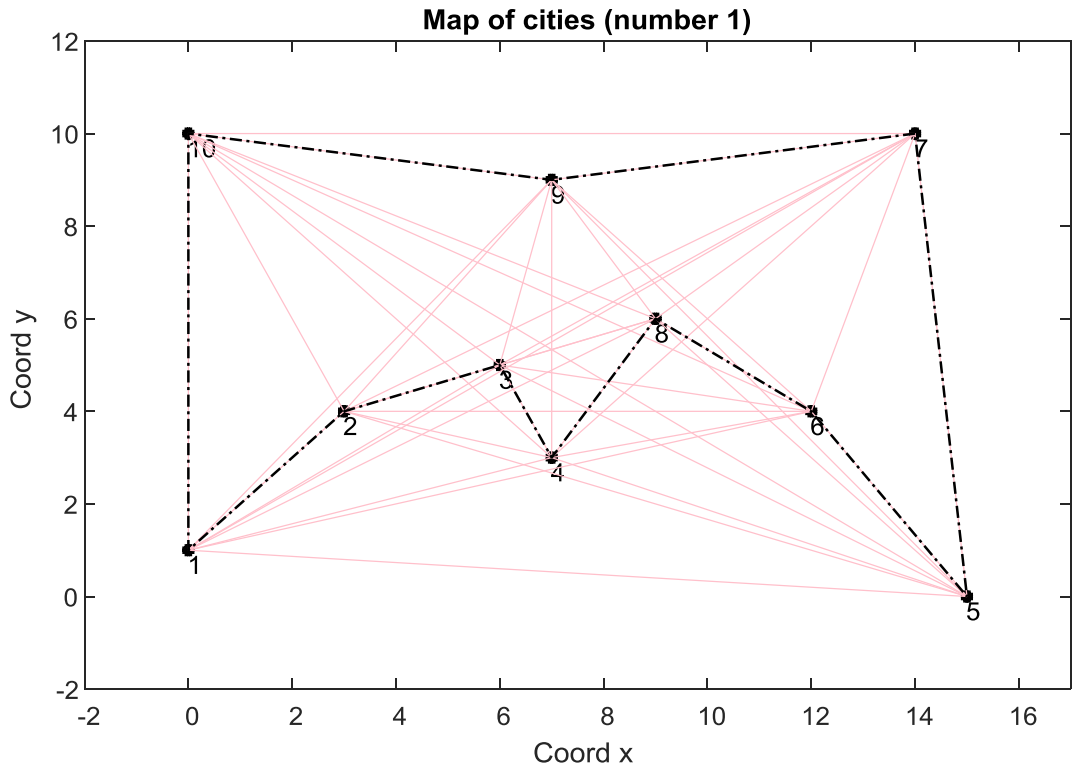

Figure 1. An example of a city network (N=10) with a TSP solution.

2.2. Results for task 2:

Table 1. Result table for task 2.

Minimal total distance for 10 trials			
Population size (P)	Crossover probability (n)	Mutation probability (p_m)	Mean of the minimal cost
100	0.5	0.1	55.5167
		0.3	55.15314
		0.5	55.3167
	0.7	0.1	55.15314
		0.3	55.09862
		0.5	55.20766
	0.9	0.1	55.26218
		0.3	55.51184
		0.5	55.09862
300	0.5	0.1	55.0441
		0.3	55.0441
		0.5	55.0441
	0.7	0.1	55.0441
		0.3	55.0441
		0.5	55.0441
	0.9	0.1	55.0441
		0.3	55.0441
		0.5	55.0441
500	0.5	0.1	55.0441
		0.3	55.0441
		0.5	55.0441
	0.7	0.1	55.0441
		0.3	55.0441
		0.5	55.0441
	0.9	0.1	55.0441
		0.3	55.0441
		0.5	55.0441

3. Conclusions

Genetic algorithms are very useful for solving optimization problems, image recognition and predicting stock market movements. In the traveling salesman problem, the main goal is to find the shortest distance between N different cities. In our algorithm, the number of cities that could be visited was 10, and the number of generations (iterations of the algorithm was 1000). The size of the population, the mutation rate and the percentage of the population included in the parental pool were variable parameters. The genetic algorithm in solving the traveling salesman problem

uses primarily randomness in the selection of the population, crossover and mutation to best reflect the populations present in nature.

Based on the obtained results, it can be concluded that the size of the population is of great importance in the case of solving the traveling salesman problem with the use of a genetic algorithm. In the case where the population size was 100, the algorithm failed to find the most optimal route each time. Since we have 10 cities, the number of possible permutations is 10! which gives over 3 million possible routes. In the case of a population of 100, we randomly choose only 100 out of these three million possible options, so it could happen that a randomly selected population would have less optimal routes, although sometimes it happens that crossing two weak individuals together gives offspring with a good genotype . If we select a larger population, it is more likely to encounter the most optimal route. It may also happen that a function stops at some local minimum and therefore may give worse results.

In the case when the population size was 300 and 500, the algorithm managed to find the shortest route for the first and each subsequent time. Even changing parameters such as the mutation rate (p_m) or the percentage of chromosomes selected for the parental pool (n) did not affect the algorithm's result. It follows that for the analyzed map of cities (number 1), the population of 100 individuals is insufficient to determine the optimal and shortest route.