БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ ИНФОРМАТИКИ И РАДИОЭЛЕКТРОНИКИ

ГиЭВ

Лабораторная работа №3

Альромхин Джорж

гр.858301

Минск 2019

**Лабораторная работа №3**

**Решение задач комбинаторной оптимизации на примере задачи коммивояжера с помощью генетического алгоритма**

**Цель задания**

Знакомство с базовыми задачами комбинаторной оптимизации, их постановками и особенностями решения с помощью ГА. Изучение метода решения задачи коммивояжера с помощью ГА. Изучение основных способов представления особей-туров и основных операторов рекомбинации для каждого представления. Реализация генетического алгоритма, решающего задачу коммивояжера.

**Постановка задачи**

Дано множество городов, заданных координатами (xi,yi) на плоскости. Стоимость перехода от одного города к другому определяется евклидовым расстоянием между ними. Решить задачу коммивояжера, т.е. построить тур обхода всех городов минимальной стоимости, с помощью генетического алгоритма. При решении использовать представление путей, оператор кроссинговера и тестовый пример, указанный в варианте. Иллюстрировать графически поиск оптимального тура.

**Вариант 2[¶](https://render.githubusercontent.com/view/ipynb?commit=c59f5de2f66d97c356a42bb015af2dd06964e58f&enc_url=68747470733a2f2f7261772e67697468756275736572636f6e74656e742e636f6d2f5374756e62612f67656e657469632d616c67732d6c6162732f633539663564653266363664393763333536613432626230313561663264643036393634653538662f6c6162332e6970796e62&nwo=Stunba%2Fgenetic-algs-labs&path=lab3.ipynb&repository_id=224357632&repository_type=Repository" \l "%D0%92%D0%B0%D1%80%D0%B8%D0%B0%D0%BD%D1%82-2)**

Упорядоченный ОК (ОХ), файл eil51.tsp.

**функция TSP: выполняет генетический алгоритм для решения задачи коммивояжера**

%xy\_coord = tsp\_readlib('eil51.tsp')

xy\_coord = [37 52; 49 49; 52 64; 20 26; 40 30; 21 47; 17 63; 31 62; 52 33; 51 21; 42 41; 31 32; 5 25; 12 42; 36 16; 52 41; 27 23; 17 33; 13 13; 57 58; 62 42; 42 57; 16 57; 8 52; 7 38; 27 68; 30 48; 43 67; 58 48; 58 27; 37 69; 38 46; 46 10; 61 33; 62 63; 63 69; 32 22; 45 35; 59 15; 5 6; 10 17; 21 10; 5 6; 10 17; 21 10; 5 64; 30 15; 39 10; 32 39; 25 32; 25 55; 48 28; 56 37; 30 40;]; % City Coordinates

POP\_SIZE = 60; % population size

NUM\_OF\_GENERATIONS = 100 ; % number of generations

[num\_of\_cities, ~] = size(xy\_coord); % get the number of cities

global\_min = Inf; % positive integers

dist\_matrix = []; %point a to b distances/cost

pop = zeros(POP\_SIZE, 54); % N by N matrix of zeroes

total\_dist = zeros(1,POP\_SIZE); %total distance of the best path

distHistory = zeros(1,NUM\_OF\_GENERATIONS); % initialize an N by N matrix of zeroes, will be used for ploting

tmp\_pop = zeros(4,num\_of\_cities); % initialize an N by N matrix of zeroes

new\_pop = zeros(POP\_SIZE,num\_of\_cities); % initialize an N by N matrix of zeroes

current\_gen = 0; %number of generations to reach global minima

gen = 0; %iterator for generations

POP\_SIZE = 4\*ceil(POP\_SIZE/4);

nPoints = size(xy\_coord,1);

display(nPoints);

a = meshgrid(1:nPoints);

dist\_matrix = reshape(sqrt(sum((xy\_coord(a,:)-xy\_coord(a',:)).^2,2))...

,nPoints,nPoints); %compute distances from each points

display(dist\_matrix);

**Создание популяции с помощью кодировки перестановок**

pop(1,:) = (1:num\_of\_cities);

for k = 2:POP\_SIZE

pop(k,:) = randperm(num\_of\_cities);

end

**Step 2: оценка пригодности: после инициализации совокупности, значение пригодности для каждого из личность подсчитывается. Значения пригодности генерируются с помощью фитнес-функция. Функция предоставляет самые большие и самые маленькие значения**

**для каждого из индивидуумов. Если человек имеет большую физическую форму значение, то результат будет лучшим решением, но если меньшее значение полученный тогда раствор получается не лучше**

figure('Name','Best Solution');

for gen = 1:NUM\_OF\_GENERATIONS

% Evaluate Each Population Member (Calculate Total Distance)

for p = 1:POP\_SIZE

tmp\_tot\_dist = dist\_matrix(pop(p,num\_of\_cities), pop(p, 1)); % temporary variable for total dstance distance

for k = 2:num\_of\_cities

tmp\_tot\_dist = tmp\_tot\_dist + dist\_matrix(pop(p,k-1),pop(p,k)); % summation of distances of pop(k)

end

total\_dist(p) = tmp\_tot\_dist; %select as elite

end

**Step 3 : Найти наименьшую стоимость, если новое вычисленное общее расстояние меньше чем последнее расстояние (global\_min), то новый глобальный минимум расстояние-это min\_dist**

[min\_dist,index] = min(total\_dist);

distHistory(gen) = min\_dist;

if min\_dist < global\_min

global\_min = min\_dist;

optimal\_route = pop(index, :);

path = optimal\_route([1 : num\_of\_cities 1]);

% Plot the Best Route

%pause(1);

plot(xy\_coord(path, 1), xy\_coord(path, 2), 'r.-');

title(sprintf('Distance = %1.4f',min\_dist));

drawnow;

current\_gen = gen;

end

**Step 4: КРОССОВЕР И МУТАЦИЯ:**

randomOrder = randperm(POP\_SIZE);

for p = 4:4:POP\_SIZE

paths = pop(randomOrder(p-3:p),:); % select parent 1 from the random population

dists = total\_dist(randomOrder(p-3:p)); %select parent 2 from the matrix of ELITE population

[~,idx] = min(dists); % get the index of the path with the min distance from the elite population

possible\_route = paths(idx,:); %get the values/permutation encoding

routeInsertionPoints = sort(ceil(num\_of\_cities\*rand(1,2)));

I = routeInsertionPoints(1);

J = routeInsertionPoints(2);

for k = 1:4 % Mutate and create a 4 new set of population

tmp\_pop(k,:) = possible\_route;

display(tmp\_pop);

display(size(tmp\_pop));

switch k

case 2

tmp\_pop(k,I:J) = tmp\_pop(k,J:-1:I); % mutate by flipping

case 3

tmp\_pop(k,[I J]) = tmp\_pop(k,[J I]); %mutate by swapping

case 4

tmp\_pop(k,I:J) = tmp\_pop(k,[I+1:J I]); % mutate by sliding

otherwise % default

end

end

new\_pop(p-3:p,:) = tmp\_pop;

%end of generation

end

pop = new\_pop; % new generation

end

Графики

figure('Name','RESULTS','Numbertitle','off');

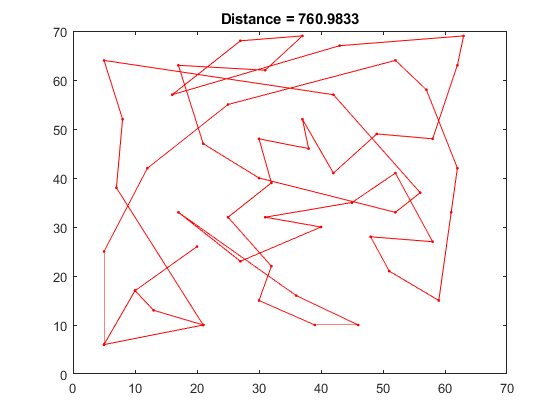
title('Best Solution History');

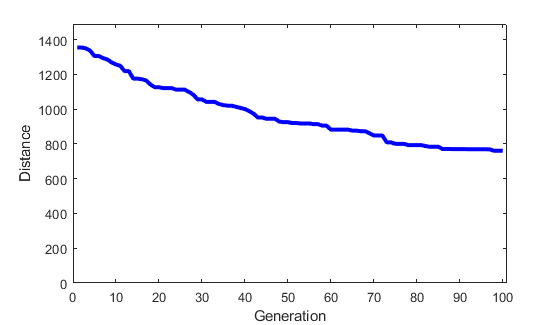
plot(distHistory,'b','LineWidth',3);

set(gca,'XLim',[0 NUM\_OF\_GENERATIONS+1],'YLim',[0 1.1\*max([1 distHistory])]);

xlabel('Generation');

ylabel('Distance');



****

## Вывод

В данной лабораторной работе реализован алгоритм нахождения приближенного решения задачи коммивояжера. Решение использует геном в виде представления путей, тип кроссинговера - упорядоченный ОК, турнирный отбор, элитарную схему сокращения популяции.