% 基于遗传算法的栅格法机器人路径规划

clc;

clear;

% 输入数据,即栅格地图.20行20列

Grid= [0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0;

0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0;

0 0 1 0 0 0 1 1 1 0 0 0 0 0 0 0 0 0 0 0;

0 0 0 0 0 0 1 1 1 0 0 0 0 0 0 0 0 0 0 0;

0 0 0 0 0 0 1 1 1 0 0 0 0 0 0 0 0 0 0 0;

0 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0;

0 1 1 1 0 0 1 1 1 0 0 0 0 0 0 0 0 0 0 0;

0 0 0 0 0 0 1 1 1 0 1 0 1 1 0 0 0 0 0 0;

0 1 1 1 0 0 0 0 0 0 1 0 1 1 0 0 0 0 0 0;

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0;

0 0 0 0 0 0 0 1 1 0 1 1 1 1 0 0 0 0 0 0;

0 0 0 0 0 0 0 1 1 0 0 0 0 0 0 0 0 0 0 0;

0 0 0 0 0 0 0 0 0 0 0 1 1 1 0 1 1 1 1 0;

0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0;

0 0 1 1 0 0 0 0 0 0 1 1 1 1 1 1 1 1 1 0;

0 0 1 1 0 0 1 1 1 0 1 0 0 0 0 0 0 0 0 0;

0 0 0 0 0 0 1 1 1 0 1 0 0 0 0 0 0 1 1 0;

0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 1 1 0;

0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0;

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0];

start\_num = 0; % 起点编号

end\_num = 399; % 终点序号

NP = 200; % 种群数量

max\_gen = 50; % 最大进化代数

pc = 0.8; % 交叉概率

pm = 0.2; % 变异概率

a = 1; % 路径长度比重

b = 7; % 路径顺滑度比重

z = 1;

new\_pop1 = {}; % 元胞数组，存放路径

[y, x] = size(Grid);

% 起点所在列（从左到右编号1.2.3...）

start\_column = mod(start\_num, x) + 1;

% 起点所在行（从上到下编号行1.2.3...）

start\_row = fix(start\_num / x) + 1; %Y = fix(X) 将 X 的每个元素朝零方向四舍五入为最近的整数

% 终点所在列、行

end\_column = mod(end\_num, x) + 1;

end\_row = fix(end\_num / x) + 1;

%% 种群初始化step1，必经节点,从起始点所在行开始往上，在每行中挑选一个自由栅格，构成必经节点

pass\_num = end\_row - start\_row + 1; %每条路径的节点个数

pop = zeros(NP, pass\_num);%生成种群数量\*节点个数的矩阵，用于存放每个个体的路径

for i = 1 : NP %每个个体（每行）循环操作：

pop(i, 1) = start\_num; %每行第一列都为起点(存入起点的编号)

j = 1;

% 此for循环用于寻找除去起点和终点所在行以外每行中的自由栅格

for row\_i = start\_row+1 : end\_row-1 %栅格的第二行到倒数第二行循环

j = j + 1;

% 存放栅格里当前行中的自由栅格序号

free = [];

for column\_i = 1 : x %从第一列到第二十列中

% 栅格对应的序号

num = (column\_i - 1) + (row\_i - 1) \* x;

% 如果该栅格为非障碍物

if Grid(row\_i, column\_i) == 0

% 把此栅格的编号加入free矩阵中

free = [free num];

end

end % 栅格一行里的自由栅格查询结束，自由栅格的编号存在了向量中

free\_num = length(free);

% 产生小于等于本行自由栅格数量的一个随机整数

index = randi(free\_num); %X = randi(imax) 返回一个介于 1 和 imax 之间的伪随机整数标量。

% 将栅格中当前行的自由栅格矩阵free中第index个栅格编号作为当前种群的第j个节点

pop(i, j) = free(index);

end %该个体的每一行的路径节点产生完成,存入了pop的第i行中

pop(i, end) = end\_num; %pop的每行第最后一列都为终点(存入终点的编号)

%% 种群初始化step2将上述必经节点联结成无间断路径

single\_new\_pop = generate\_continuous\_path(pop(i, :), Grid, x);

if ~isempty(single\_new\_pop)%如果这一行种群的路径不是空的，将这行路径存入元胞数组中。

new\_pop1(z, 1) = {single\_new\_pop};

z = z + 1;

end

end

%% 计算初始化种群的适应度

% 计算路径长度

path\_value = cal\_path\_value(new\_pop1, x);

% 计算路径平滑度

path\_smooth = cal\_path\_smooth(new\_pop1, x);

fit\_value = a .\* path\_value .^ -1 + b .\* path\_smooth .^ -1;

mean\_path\_value = zeros(1, max\_gen);

min\_path\_value = zeros(1, max\_gen);

%% 循环迭代操作

for i = 1 : max\_gen

% 选择操作

new\_pop2 = selection(new\_pop1, fit\_value);

% 交叉操作

new\_pop2 = crossover(new\_pop2, pc);

% 变异操作

new\_pop2 = mutation(new\_pop2, pm, Grid, x);

% 更新种群

new\_pop1 = new\_pop2;

% 计算适应度值

% 计算路径长度

path\_value = cal\_path\_value(new\_pop1, x)

% 计算路径平滑度

path\_smooth = cal\_path\_smooth(new\_pop1, x)

fit\_value = a .\* path\_value .^ -1 + b .\* path\_smooth .^ -1

mean\_path\_value(1, i) = mean(path\_value);

[~, m] = max(fit\_value);

min\_path\_value(1, i) = path\_value(1, m);

end

%% 画每次迭代平均路径长度和最优路径长度图

figure(1)

plot(1:max\_gen, mean\_path\_value, 'r')

hold on;

title(['a = ', num2str(a)', '，b = ',num2str(b)','的优化曲线图']);

xlabel('迭代次数');

ylabel('路径长度');

plot(1:max\_gen, min\_path\_value, 'b')

legend('平均路径长度', '最优路径长度');

min\_path\_value(1, end)

% 在地图上画路径

[~, min\_index] = max(fit\_value);

min\_path = new\_pop1{min\_index, 1};

figure(2)

hold on;

title(['a = ', num2str(a)', '，b = ',num2str(b)','遗传算法机器人运动轨迹']);

xlabel('坐标x');

ylabel('坐标y');

DrawMap(Grid);

[~, min\_path\_num] = size(min\_path);

for i = 1:min\_path\_num

% 路径点所在列（从左到右编号1.2.3...）

x\_min\_path(1, i) = mod(min\_path(1, i), x) + 1;

% 路径点所在行（从上到下编号行1.2.3...）

y\_min\_path(1, i) = fix(min\_path(1, i) / x) + 1;

end

hold on;

plot(x\_min\_path, y\_min\_path, 'r')