基于遗传算法的校园最短游览路线选择

姓名：唐川淇 学号：1131190111

# 问题重述



江南开学府，万顷湖波扬，屏障九龙好，山高水又长。江南大学风景如画，在江南大学里有许多值得打卡的地点，那么如何规划路线使得可以参观所有的地点的同时所花费的路程最短呢？

# 问题分析



首先选择出十二个打卡地点，并计算出他们的坐标位置，如下图。校内的路线大多为东西和南北走向，因此使用公式计算出任意两点之间的距离。根据遗传算法，可以初始化生成一些路线，随着遗传交叉变异逐渐得到更距离的解，最终可以得到最终的最短距离的游览路线。



# 模型的建立与求解

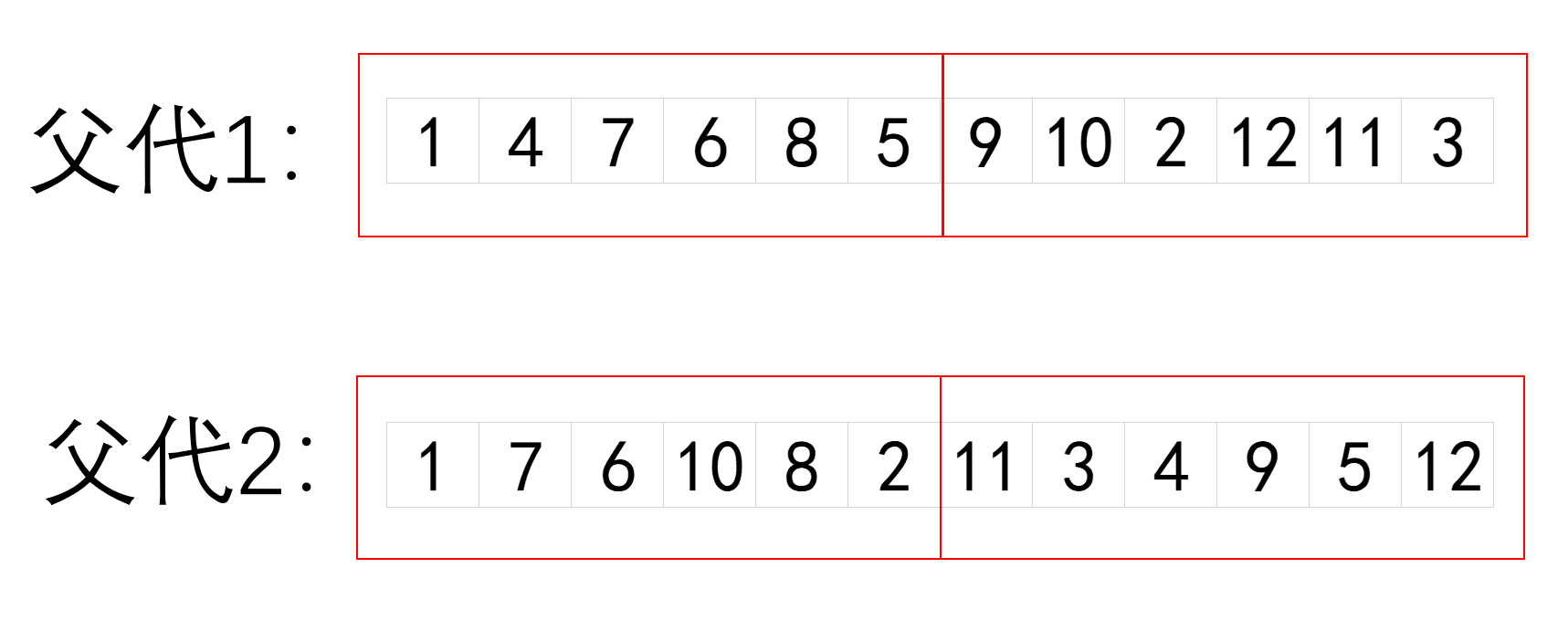
## 初始化

初始化种群规模，最大迭代次数，变异系数等参数。其中适应度设置为

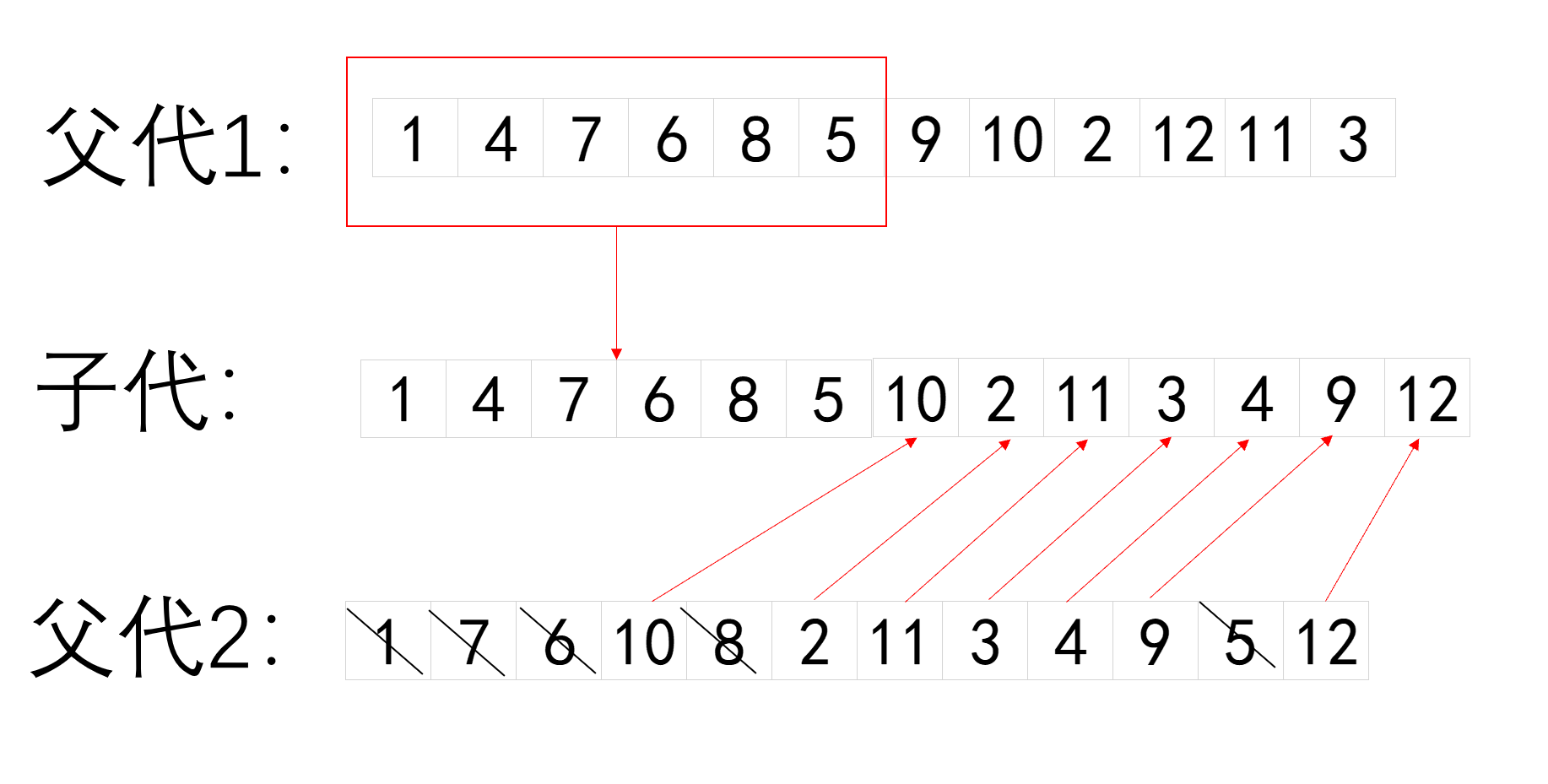
## 轮盘赌选择父代染色体

有一个轮盘，现在我们将它分割成 若干个部分，代表我们总体中染色体的个数。每条染色体在轮盘上占有的区域面积将根据适应度分数成比例表达出来。现在，这个轮盘开始旋转，我们将被固定的指针指到的那片区域选为第一个亲本。然后，对于第二个亲本，我们进行同样的操作。最终选出父代。

## 交叉



选出的父代如上图所示，将两父代都拆分成两部分，如红色方框框出所示。



首先将父代1的前半部分取出得到子代的前半部分，之后从从父代2中将子代中出现的数字划掉，将父代2剩余的部分拼接到后面，最终的得到子代。同理也可将父代2的前半部分取出，将父代1去重后拼接到父代2的后面得到子代。

## 变异

|  |  |
| --- | --- |
|  | 变异则按照一定概率随机交换序列中的数字。 |

# 模型的结果

利用MATLAB计算得到如下游览路线：



附录

|  |
| --- |
| 附录 |
|  |
| function f = genetic\_algorithm(D, mute, Pm)  clear;  if ~exist('mute', 'var')  mute = 0; % 是否显示各种提示信息  end  if ~exist('Pm', 'var')  Pm = 0.5; % 变异概率，越大收敛越慢但是解一般越好  end  if ~exist('D', 'var')  data=xlsread('坐标.xlsx');  C=data;  % D为城市间的距离矩阵  R=zeros(12);  for i=1:12  for j=1:12  R(i,j)=abs(data(i,1)-data(j,1))+abs(data(i,2)-data(j,2));  end  end  D = num2str(R,'%10.2f');  %disp(D);  end  rng(1);  n = size(D, 1);  N = 500; % 群体规模  TOL = 200; % 最大容忍次数(连续TOL次rate不上升，或找不到更优解，则停止迭代)  solutions = zeros(N, n);  fs = zeros(N, 1);  for i = 1:N  solutions(i, :) = [1, randperm(n-1) + 1]; % 生成N个解，假定从1开始  fs(i) = TSP\_distance(D, solutions(i, :));  end  Pu = max(fs) - fs + 1;  P = Pu/sum(Pu);  cumP = cumsum(P);  best = min(fs);  avg = mean(fs);  rate = best/avg;  if ~mute  disp('初始解的群体中最短的路径长度为：');  disp(best);  disp('初始解的群体中平均路径长度为：');  disp(avg);  end  tol = 0;  count = 0;  while 1  count = count + 1;  if ~mute  fprintf('当前第%d次迭代\n', count);  end  parents = zeros(size(solutions));  for i = 1:N % 使用轮盘赌的方式选出父代的染色体  index = sum(cumP <= rand) + 1;  parents(i, :) = solutions(index, :);  end  new\_solutions = zeros(size(solutions));  assert(mod(N, 2) == 0);  for i = 1:N/2 % 交配操作；这里默认N为偶数，每两个父代一起产生两个子代  % 产生的子代1取父代1的前一半染色体，后一半则由父代2提供；同理于子代2  p1 = parents(2\*i-1, :);  p2 = parents(2\*i, :);  middle = ceil(n/2);  s1 = p1(1:middle);  res1 = setdiff(p2, s1, 'stable');  s1 = [s1, res1];  s2 = p2(1:middle);  res2 = setdiff(p1, s2, 'stable');  s2 = [s2, res2];  new\_solutions(2\*i-1, :) = s1;  new\_solutions(2\*i, :) = s2;  end  for i = 1:N % 变异操作；变异的方式为随机取两个顺序交换  if rand < Pm  temp = randperm(n-1) + 1;  k = temp(1);  new\_solutions(i, [1, k]) = new\_solutions(i, [k, 1]);  end  end  % 至此，新的种群已经生成完毕，代替旧种群后开始新一轮的计算  solutions = new\_solutions;  for i = 1:N  fs(i) = TSP\_distance(D, solutions(i, :));  end  Pu = max(fs) - fs + 1;  P = Pu/sum(Pu);  cumP = cumsum(P);  best\_new = min(fs);  avg = mean(fs);  rate\_new = best\_new/avg;  if ~mute  disp('最短的路径长度为：');  disp(best\_new);  disp('平均的路径长度为：');  disp(avg);  end  tol = tol + 1;  if best\_new < best || rate\_new > rate  best = best\_new;  rate = rate\_new;  tol = 0;  end  if tol >= TOL  break  end  if count > 5000  break  end  end  [f, index] = min(fs);  solution = solutions(index, :);  if ~mute  fprintf('最后搜索得到的最优路径为：\n');  disp(solution);  disp('路径长度为：');  disp(f);  end  function f = TSP\_distance(D, solution)  % 本函数计算给定solution的距离，其中距离由D给出。  n = numel(solution);  sum = 0;  for i = 1:n-1  sum = sum + D(solution(i), solution(i+1));  end  sum = sum + D(solution(n), solution(1));  f = sum; |