数学建模常用算法 II

数模国赛临门一脚冲刺课程

周吕文

超级数学建模大俱乐部

2018 年 9 月 8 日


招码听课


超级数模

Notes

单服务台

开始服务, 到达, 离开时刻和服务, 等待时长的关系

$$\text{服务时刻}(i) = \max\left\{\text{到达时刻}(i), \text{离开时刻}(i-1)\right\}$$

$$\text{离开时刻}(i) = \text{服务时刻}(i) + \text{服务时长}(i)$$

$$\text{等待时长}(i) = \text{离开时刻}(i) - \text{到达时刻}(i)$$

Notes

多服务台

开始服务, 到达, 离开时刻和服务, 等待时长的关系

$$\text{服务时刻}(i) = \max\left\{\text{到达时刻}(i), \min\{\text{服务台空闲时刻}\}\right\}$$

$$\text{所使用服务台}(i) = k, \quad \text{其中 } k \text{ 使 服务台空闲时刻}(k) = \min$$

$$\text{离开时刻}(i) = \text{服务时刻}(i) + \text{服务时长}(i)$$

$$\text{服务台空闲时刻}(k) = \text{离开时刻}(i)$$

$$\text{等待时长}(i) = \text{离开时刻}(i) - \text{到达时刻}(i)$$

Notes

自动取款机问题

问题

银行计划安置取款机, A 机价格和平均服务率都是 B 机的 2 倍. 应购置 1 台 A 机还是 2 台 B 机?

顾客平均每分钟到达 1 位, A 型机的平均服务时间为 0.9, B 型机为 1.8 分钟, 顾客到达间隔和服务时间都服从指数分布.

Notes

单服务台

mm1.m

```
01 n = 100000;                          % 模拟顾客总数
02 mu = 1; muA = 0.9;                   % 到达率和服务率
03 tarr = cumsum(exprnd(mu,1,n));       % 到达时刻
04 tsrv = exprnd(muA,1,n);              % 服务时长
05 tsta = zeros(1,n);                   % 初始化服务时刻
06 tlea = zeros(1,n);                   % 初始化离开时刻
07 twat = zeros(1,n);                   % 初始化等待时长
08 tsta(1) = tarr(1);                   % 首位顾客服务时刻=到达时刻
09 tlea(1) = tsta(1) + tsrv(1);         % 首位顾客离开时刻
10 twat(1) = tlea(1) - tarr(1);         % 首位顾客等待时长=0
11 for i = 2:n
12     % 服务时刻 = max{到达时刻, 上一个顾客离开时刻}
13     tsta(i) = max(tarr(i),lea(i-1));
14     tlea(i) = tsta(i) + tsrv(i);% 离开时刻=服务时刻+服务时长
15     twat(i) = tlea(i) - tarr(i);% 等待时长=离开时刻-到达时刻
16 end
```


Notes

两服务台

mm2.m

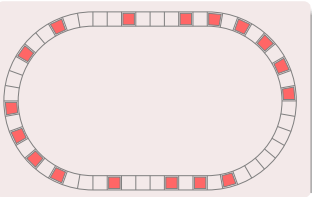
```
01 n = 100000;                          % 模拟顾客总数
02 mu = 1; muB = 1.8;                   % 到达率和服务率
03 tarr = cumsum(exprnd(mu,1,n));       % 到达时刻
04 tsrv = exprnd(muB,1,n);              % 服务时长
05 tsta = zeros(1,n); tlea = zeros(1,n);% 初始化服务/离开时刻
06 twat = zeros(1,n);                   % 初始化等待时长
07 last = [0 0];                        % 初始化服务台结束服务时刻
08 for i = 1:n
09     [minval, k] = min(last);  % 找出最快结束服务的服务台时刻
10     tsta(i) = max(tarr(i),minval);% 服务时刻
11     tlea(i) = tsta(i) + tsrv(i);  % 离开时刻
12     last(k) = tlea(i);            % 服务台结束服务时刻
13     twat(i) = tlea(i) - tarr(i);  % 等待时长
14 end
```


Notes

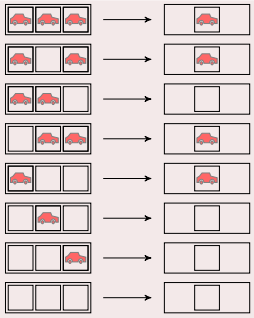
交通规则

定义

元胞分布于一维线性网格上.
元胞仅具有车和空两种状态.
元胞状态由周围两邻居决定.

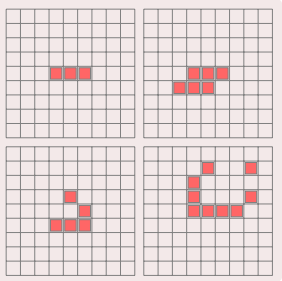


规则



Notes

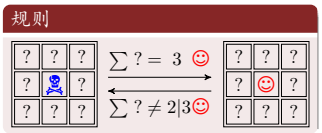
生命游戏



定义

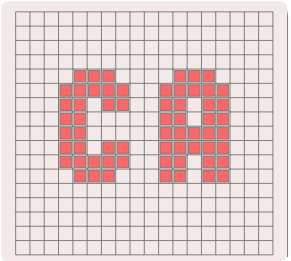
元胞分布于二维方型网格上.
元胞仅具有生和死两种状态.
元胞状态由周围八邻居决定.

规则



Notes

什么是元胞自动机



元胞自动机是离散的动力学系统

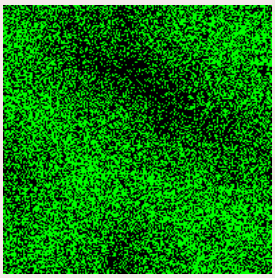
离散的系统：元胞是定义在有限的时间和空间上的，并且元胞的状态是有限.

动力学系统：元胞自动机的举止行为具有动力学特征.

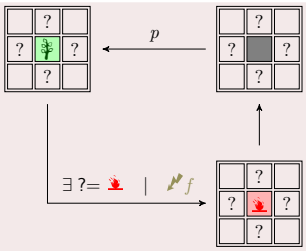
简单与复杂：元胞自动机用简单规则控制相互作用的元胞模拟复杂世界.

Notes

森林火灾：规则



规则



Notes

森林火灾：理论分析

三种状态元胞密度关系

$$\rho_f + \rho_e + \rho_t = 1$$

系统稳定条件

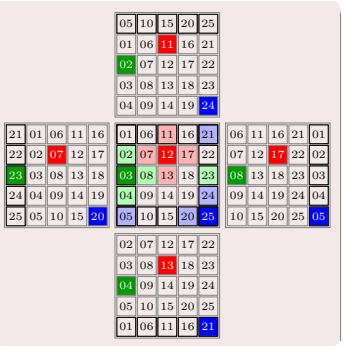
$$\rho_f = p\rho_e$$

树生长与火灾蔓延的时间尺度分离条件

$$f \ll p \ll T_{\text{smax}}$$

Notes

森林火灾：程序实现



上邻居

```
U = C([5, 1:4], :)
```

下邻居

```
D = C([2:5, 1], :)
```

左邻居

```
L = C(:, [5, 1:4])
```

右邻居

```
R = C(:, [2:5, 1])
```


Notes

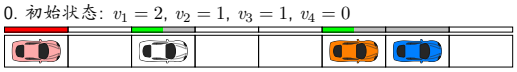
森林火灾：程序实现

```
01 n = 300;                          % 定义表示森林的矩阵大小
02 Plight  = .000005; Pgrowth = .01;% 定义闪电和生长的概率
03 UL = [n 1:n-1];  DR = [2:n 1];   % 定义上左，下右邻居
04 veg=zeros(n,n);                   % 初始化表示森林的矩阵
05 imh = image(cat(3,veg,veg,veg)); % 可示化表示森林的矩阵
06 % veg = {empty=0 burning=1 green=2}
07 for i=1:3000                      % 主循环开始
08     sum =          (veg(UL,:)==1) + ...
09          (veg(:,UL)==1)    +    (veg(:,DR)==1) + ...
10                     (veg(DR,:)==1);% 求上下左右四个邻居和
11 % 根据规则更新森林矩阵：树 = 树 - 着火的树 + 新生的树
12     veg = 2*(veg==2) - ...
13           ( (veg==2) & (sum>0|(rand(n,n)<Plight)) ) + ...
14           2*((veg==0) & rand(n,n)<Pgrowth) ;
15     set(imh, 'cdata', cat(3,(veg==1),(veg==2),zeros(n)) )
16     drawnow                       %可示化表示森林的矩阵
17 end                               %主循环结束
```

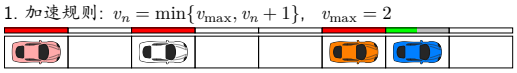

Notes

交通模拟：规则

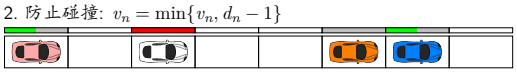
0. 初始状态：$v_1 = 2,\ v_2 = 1,\ v_3 = 1,\ v_4 = 0$



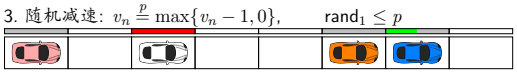
1. 加速规则：$v_n = \min\{v_{\max}, v_n + 1\},\quad v_{\max} = 2$



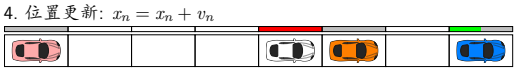
2. 防止碰撞：$v_n = \min\{v_n, d_n - 1\}$



3. 随机减速：$v_n \overset{p}{=} \max\{v_n - 1, 0\},\qquad \mathsf{rand}_1 \le p$



4. 位置更新：$x_n = x_n + v_n$



Notes

交通模拟：理论分析

由车道长，车数，最大速度
$L = 36,\ N = 12,\ v_{\max} = 5,$
容易求得平均密度和间距
$\rho = \frac{N}{L} = \frac{1}{3},\ d = \frac{L-N}{N} = 2$
如何求平均速度和车流量？

临界密度

$$\rho_c = \frac{1}{v_{\max} + 1} = \frac{1}{5+1}$$

平均速度

$$v = \min\{v_{\max}, d\} = 2$$

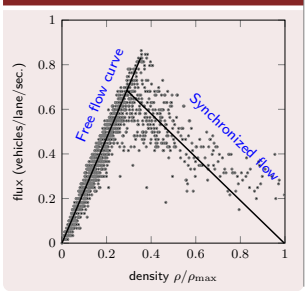
车流量

$$J = \min\{\rho v_{\max}, \rho d\} = 2/3$$
$$= \min\{\rho v_{\max}, 1 - \rho\}$$

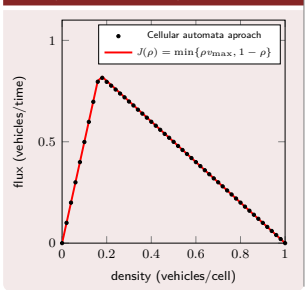
Notes

交通模拟：密度与流量

观测数据



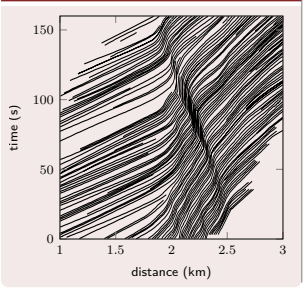
理论值及 CA 预测值



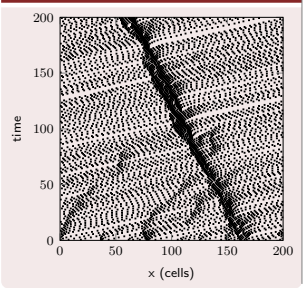
Notes

交通模拟：时空轨迹

航拍数据



CA 计算结果 ($p_{\text{brake}} = 0.15$)



Notes

交通模拟：程序实现

ns.m

```
01 function flux = ns(rho,p,L,tmax)% rho=0.2;p=0.2;L=72;tmax=72
02 ncar = round(L*rho);             % ncar=L*rho
03 x = sort(randperm(L, ncar));     % 1:ncar中随机不重复的L个数
04 vmax = 5;                        % 最大速度
05 v = vmax * ones(1, ncar);        % 初始化所有车速度为vmax
06 for t = 1:tmax
07     v = min(v+1, vmax);             % 加速规则
08     gaps = gaplength(x,L,ncar);
09     v = min(v, gaps-1);             % 防止碰撞
10     v = max(v- (rand(1,ncar)<p), 0);% 随机减速 binornd
11     x = x + v;                      % 位置更新
12     x(x>L) = x(x>L) - L;            % 周期边界
13     flux = flux + sum(v)/L;         % 空间平均
14 end
15 flux = flux / tmax;              % 时间平均
```

```
16 function gaps = gaplength(x, L, ncar)
17 gaps = zeros(1, ncar);
18 gaps = x([2:end 1]) -x;          % d(i) = x(i+1)-x(i)
19 gaps(gaps<=0) = gaps(gaps<=0)+L;% d(i) = d(i) + L, if d(i)<0
```

Notes

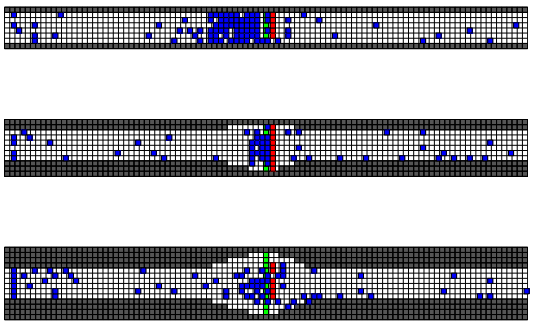
交通模拟：交通问题

MCM2005B

MCM2009A

Notes

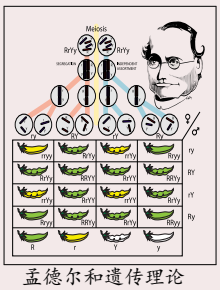
交通模拟：收费亭最优数量



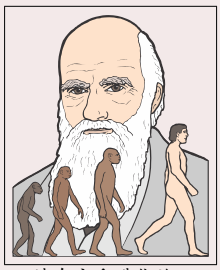
Notes

遗传算法的启源

生物中的遗传与进化理论



孟德尔和遗传理论



达尔文和进化论

Notes

基本概念

种群

染色体 1 | 1 2 3 4 5

染色体 2 | 2 4 3 8 9

⋮

染色体 n | 7 8 3 6 0

基因

基因 (Gene)
染色体上的一个单元，解中的一个参数.

染色体 (Chromosome)
由一组基因构成，问题可能的一个解.

种群 (Population)
由一系列染色体组成的一个集合.

Notes

遗传算法基本思想

遗传算法伪代码

1: set initial generation $k = 0$
2: probability of mutation $= \alpha$
3: probability of performing crossover $= \beta$
4: construct a population of n randomly-generated individuals P_k;
5: **while** not termination **do**
6: evaluate: compute *fitness*(i) for each individuals in P_k
7: select: select m members of P_k insert into P_{k+1}.
8: crossover: produce αm children by crossover and insert into P_{k+1}
9: mutate: produce βm children by mutate and insert into P_{k+1}
10: update generation: $k = k + 1$
11: **end while**
12: **reture** the fittest individual from P_{last}

Notes

编码

遗传算法中，首要问题就是如何对解进行编码 (解的形式). 编码影响到交叉，变异等运算，很大程度上决定了遗传进化的效率.

十进制	二进制	格雷码
0	0000	0000
1	0001	0001
2	0010	0011
3	0011	0010
4	0100	0110
5	0101	0111
6	0110	0101
7	0111	0100
8	1000	1100
9	1001	1101

二进制编码: 每个基因值为符号 0 和 1 所组成的二值制数.

格雷编码: 与二进制编码类似，连续两个整数所对应编码仅一码之差.

实数编码: 每个基因值用某一范围内的一个实数来表示.

符号编码: 染色体编码串中的基因值取自一个无数值含义，而只有代码含义的符号集.

Notes

适应度函数

适应度函数也称评价函数，是根据目标函数确定的用于区分群体中个体好坏的标准．适应度函数值的大小是对个体的优胜劣汰的依据．

通常适应度函数可以由目标函数直接或间接改造得到．比如，目标函数，或目标函数的倒数/相反数经常被直接用作适应度函数．

一般情况下适应度是非负的，并且总是希望适应度越大越好(适应度值与解的优劣成反比例)．

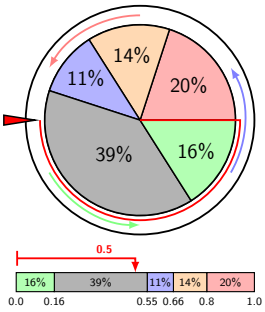
比较好的适应度函数应：单值，连续，非负，最大化．

适应度函数不应过于复杂，越简单越好，以便于计算机的快速计算．

Notes

选择

选择运算的使用是对个体进行优胜劣汰：从父代群体中选取一些适应度高个体，遗传到下一代群体．



轮盘赌：又称比例选择算子，个体 i 被选中的概率 p_i 与其适应度成正比．

$$p_i = f_i / \sum_{j=1}^{N} f_j$$

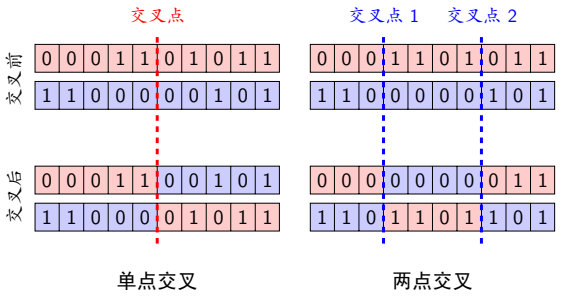
两两竞争：从父代中随机地选取两个个体，比较适应值，保存优秀个体，淘汰较差的个体．

排序选择：根据各个体的适应度大小进行排序，然后基于所排序号进行选择．

Notes

交叉

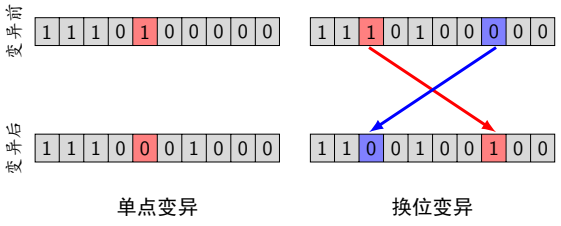
交叉运算，是指对两个相互配对的染色体依据交叉概率按某种方式相互交换其部分基因，从而形成两个新的个体．



Notes

变异

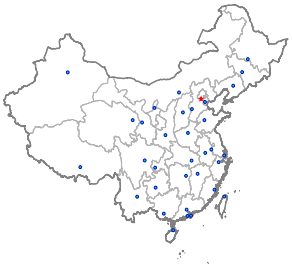
变异操作对群体中的个体的某些基因座上的基因值作变动，模拟生物在繁殖过程，新产生的染色体中的基因会以一定的概率出错．



Notes

TSP 问题的遗传算法求解

已知中国 34 个省会城市 (包括直辖市) 的经纬度, 要求从北京出发, 游遍 34 个城市, 最后回到北京. 用遗传算法求最短路径.



如何对问题的解编码?

$[1, \cdots, i, \cdots, j, \cdots, n, 1]$

如何构造适应度函数?

$1/\sum_{i=1}^{n} \mathrm{dist}(S_{(i)}, S_{(i+1)})$

如何设计选择运算?

两两竞争, 轮盘赌...

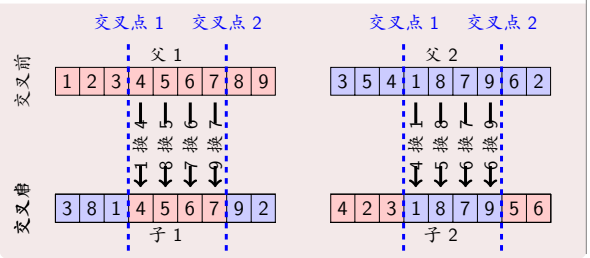
如何设计交叉运算?

如何设计变异运算?

Notes

TSP 问题的遗传算法求解

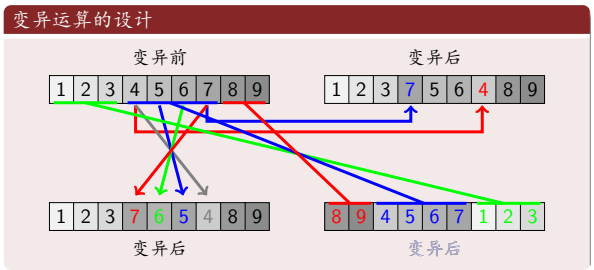
交叉运算的设计



Notes

TSP 问题的遗传算法求解

变异运算的设计



Notes

TSP 问题的遗传算法求解

主程序 MatLab 代码

```
01 popSize = 100;                    % 种群规模
02 max_generation = 1000;            % 初始化最大种群代数
03 Pmutation = 0.16;                 % 变异概率
04 for i = 1:popSize                 % 初始化种群
05    pop(i,:) =  randperm(numberofcities);
06 end
07 for generation = 1:max_generation % 主循环开始
08     fitness = 1/totaldistance(pop,dis);% 计算距离(适应度)
09     [maxfit, bestID] = max(fitness);
10     bestPop = pop(bestID, :);     % 找出精英
11     pop = select(pop,fitness,popSize,'competition');% 选择
12     pop = crossover(pop);         % 交叉
13     pop = mutate(pop,Pmutation);  % 变异
14     pop = [bestPop; pop];         % 精英保护
15 end                               % 主循环开始
16 popDist = total_distance(pop,dis);% 计算距离(适应度)
17 [minDist, index] = min(popDist);  % 找出最短距离
18 optRoute = pop(index,:);          % 找出最短距离对就的路径
```

Notes

TSP 问题的遗传算法求解

选择操作函数 select

```
01 function popselect = select(pop, fitness, nselect, method)
02 popSize = size(pop,1);
03 switch method
04     case 'roulette'                % 轮盘赌
05         p=fitness/sum(fitness);    % 选中概率 [0.2 0.3 0.5]
06         cump=cumsum(p);            % 概率累加 [0.2 0.5 1.0]
07         % 利用插值: yi = 线性插值(x, y, xi)
08         I = interp1([0 cump],1:(popSize+1), ...
09                     rand(1,nselected),'linear');
10         I = floor(I);
11     case 'competition'             % 两两竞争
12         i1 = ceil( popSize*rand(1,nselected) );
13         i2 = ceil( popSize*rand(1,nselected) );
14         I = i1.*( fitness(i1)>=fitness(i2) ) + ...
15             i2.*( fitness(i1)< fitness(i2) );
16 end
17 popselect = pop(I);
```

Notes

TSP 问题的遗传算法求解

交叉操作函数 crossover

```
01 function children = crossover(parents)
02 [popSize, numberofcities] = size(parents);
03 children = parents;                % 初始化子代
04 for i = 1:2:popSize                % 交叉开始
05     parent1 = parents(i+0,:);  child1 = parent1;
06     parent2 = parents(i+1,:);  child2 = parent2;
07     InsertPoints = ceil(numberofcities*rand(1,2));% 交叉点
08     for j = min(InsertPoints):max(InsertPoints)
09         if parent1(j)~=parent2(j) % 如果对应位置不重复
10             child1(child1==parent2(j)) = child1(j);
11             child1(j) = child2(j);
12             child2(child2==parent1(j)) = child2(j);
13             child2(j) = child1(j);
14         end
15     end
16     children(i+0,:) = child1; children(i+1,:) = child2;
17 end                               % 交叉结束
```

Notes

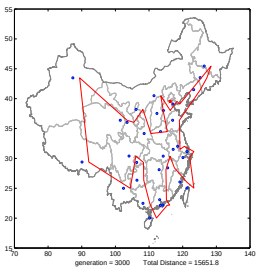
TSP 问题的遗传算法求解

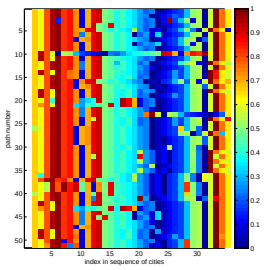
变异操作函数 mutate

```
01 function children = mutation(parents, probmutation)
02 [popSize, numberofcities] = size(parents);
03 children = parents;                % 初始化子代
04 for k=1:popSize                    % 变异开始
05     if rand < probmutation         % 以一定概率变异
06        InsertPoints = ceil(numberofcities*rand(1,2));
07        I = min(InsertPoints); J = max(InsertPoints)1
08        switch ceil(rand*4)         % swap, slide, flip
09            case 1    % [1 2 3 4 5 6 7] -> [1 5 3 4 2 6 7]
10              children(k,[I J]) = parents(k,[J I]);
11            case 2    % [1 2 3 4 5 6 7] -> [1 3 4 5 2 6 7]
12              children(k,I:J) = parents(k,[I+1:J I]);
13            otherwise % [1 2 3 4 5 6 7] -> [1 5 4 3 2 6 7]
14              children(k,I:J) = parents(k,J:-1:I);
15        end
16     end
17 end                                % 变异结束
```

Notes

TSP 问题的遗传算法结果





Notes

Thank You!!!

Notes

Notes

Notes

Notes