数学建模常用算法 I

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

提要

问题：对女星的评价

加权平均：评分和权重难以估计

$$p = \sum_{i=1}^{n} w_i p_i$$

评分和权重

	权重	苍井	小泽
颜值	0.3	95	85
身材	0.3	90	95
声音	0.2	82	85
演技	0.2	85	90

评分

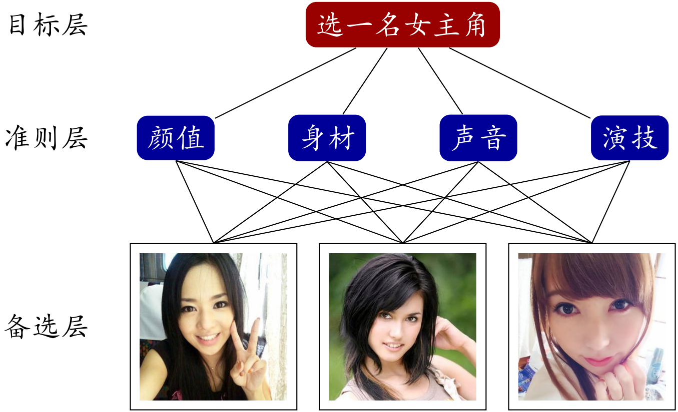
$$p_{苍} = 0.3 \times 95 + 0.3 \times 90 + 0.2 \times 82 + 0.2 \times 85$$

$$p_{小} = 0.3 \times 85 + 0.3 \times 95 + 0.2 \times 85 + 0.2 \times 90$$

wetavg.m

```
Wi = [0.3 0.3 0.2 0.2];
Pi = [ 95  90  82  85 ; 85  95  85  90 ]; % sum(Wi.*Pi,2)
P = Wi * Pi'  % = [88.9   89.0]
```

问题：对女星的评价



模型：构造判断矩阵

准则

颜值 C_1、身材 C_2、声音 C_3、演技 C_4

两两比较：C_i 相对于 C_j 的重要程度

$$a_{i,j} = \frac{C_i}{C_j} \in \{1, 2, 3, \cdots, 9\}$$

判断矩阵

$$A = \begin{bmatrix} 1/1 & 2/1 & 5/1 & 3/1 \\ 1/2 & 1/1 & 3/1 & 1/2 \\ 1/5 & 1/3 & 1/1 & 1/4 \\ 1/3 & 2/1 & 4/1 & 1/1 \end{bmatrix}$$

- $A_{14} = 3/1$ 表示颜值比演技稍重要
- $A_{13} = 5/1$ 表示颜值比声音明显重要

模型：构造判断矩阵

准则

颜值 B_1、身材 B_2、声音 B_3、演技 B_4

两两比较：B_i 相对于 B_j 的重要程度

$$a_{i,j} = \frac{B_i}{B_j} \in \{1, 2, 3, \cdots, 9\}$$

判断矩阵

$$A = \begin{bmatrix} 1/1 & 2/1 & 5/1 & 3/1 \\ 1/2 & 1/1 & 3/1 & 1/2 \\ 1/5 & 1/3 & 1/1 & 1/4 \\ 1/3 & 2/1 & 4/1 & 1/1 \end{bmatrix}$$

- $a_{14} = 3$：C_1 比 C_4 稍重要；$a_{13} = 5$：C_1 比 C_3 明显重要
- 若比较结果前后完全一致：$a_{ij}a_{jk} = a_{ik}$

模型：一致性检验

- 若 $A \gg C$ 且 $B \gtrsim C$, 则 $A > B$

一致性指标 CI、一致性比例 CR、平均随机一致性指标 RI

$$CI = \frac{\lambda - n}{n-1}, \qquad CR = \frac{CI}{RI(n)} < 0.1$$

n	1	2	3	4	5	6	7	8	9
RI	0	0	0.58	0.90	1.12	1.24	1.32	1.41	1.45

AHP.m

```
A = [1/1   2/1   5/1   3/1      % 判断矩阵
      1/2   1/1   3/1   1/2
      1/5   1/3   1/1   1/4
      1/3   2/1   4/1   1/1];
[V, D] = eig(A);                % 计算特征向量V和特征值D: A*V=V*D
[lamda, i] = max(diag(D));      % 最大特征值lambda及其位置i
CI = (lambda-4) / (4-1);        % 一致性指标
CR = CI / 0.9                   % 一致性比例 = 0.0512
```

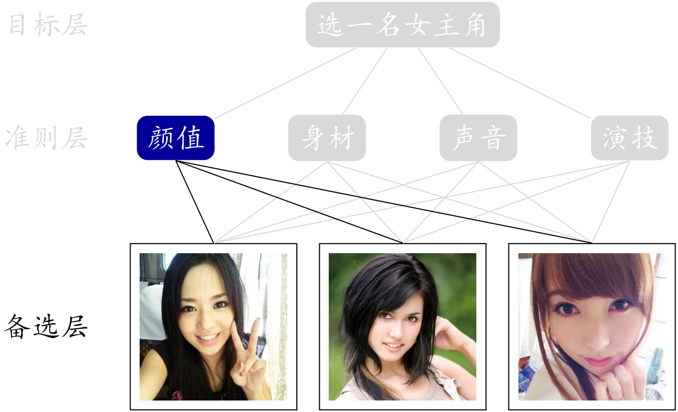
模型：层次单排序

- 对于上一层某因素而言，本层次各因素的重要性的排序。
- 上一层次某因素相对重要性：判断矩阵 A 对应于最大特征值 $\lambda_{\max}$ 的特征向量 W。

AHP.m

```
A = [1/1   2/1   5/1   3/1      % 判断矩阵
      1/2   1/1   3/1   1/2
      1/5   1/3   1/1   1/4
      1/3   2/1   4/1   1/1];
[V, D] = eig(A);                % 计算特征向量V和特征值D: A*V=V*D
[lamda, i] = max(diag(D));      % 最大特征值lambda及其位置i
W = V(:,i);                     % 最大特征值对应的特征向量
w = W/sum(W)                    % 归一化 =[0.48 0.19 0.07 0.26]'
```

模型：层次总排序



程序：层次总排序

ahpactor.m

```
A = [1/1  2/1  5/1  3/1; 1/2  1/1  3/1  1/2
     1/5  1/3  1/1  1/4; 1/3  2/1  4/1  1/1];
[w, CR] = aph(A);

A1 = [1/1 1/2 3/1; 2/1 1/1 5/1; 1/3 1/5 1/1];   % 颜值
[w1, CR1] = aph(A1); ...
P = [w1 w2 w3 w4] * w
```

AHP.m

```
function [w, CR] = AHP(A)
RI = [ 0.00 0.00 0.58 0.90 1.12 1.24 1.32 1.41 1.45];
n = size(A,1);
[V, D] = eig(A);
[lamda, i] = max(diag(D));
CI = (lambda-n) / (n-1);
CR = CI / RI(n);
W = V(:,i); w = W/sum(W);
```

拟合：线性和非线性拟合

Command Window

f_x >>

拟合：线性和非线性拟合

Command Window

```
>> x = [1.0, 1.5, 2.0, 2.5, 3.0]';
fx >>
```

拟合：线性和非线性拟合

Command Window

```
>> x = [1.0, 1.5, 2.0, 2.5, 3.0]';
>> y = [0.9, 1.7, 2.2, 2.6, 3.0]';
fx >>
```

拟合：线性和非线性拟合

Command Window

```
>> x = [1.0, 1.5, 2.0, 2.5, 3.0]';
>> y = [0.9, 1.7, 2.2, 2.6, 3.0]';
>> a = polyfit(x,y,1)

a =

      1.0200    0.0400
```

拟合：线性和非线性拟合

Command Window

```
>> x = [1.0, 1.5, 2.0, 2.5, 3.0]';
>> y = [0.9, 1.7, 2.2, 2.6, 3.0]';
>> a = polyfit(x,y,1)

a =

      1.0200    0.0400

fx >>
```

拟合：线性和非线性拟合

Command Window

```
>> x = [1.0, 1.5, 2.0, 2.5, 3.0]';
>> y = [0.9, 1.7, 2.2, 2.6, 3.0]';
>> a = polyfit(x,y,1)

a =

      1.0200    0.0400

>> xi = 1:0.1:3;
fx >>
```

拟合：线性和非线性拟合

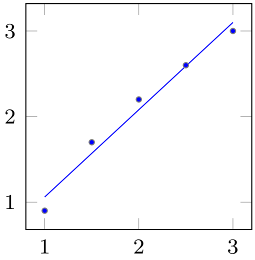
Command Window

```
>> x = [1.0, 1.5, 2.0, 2.5, 3.0]';
>> y = [0.9, 1.7, 2.2, 2.6, 3.0]';
>> a = polyfit(x,y,1)

a =

      1.0200     0.0400

>> xi = 1:0.1:3;
>> yi = polyval(a,xi);
fx >>
```

拟合：线性和非线性拟合

Command Window

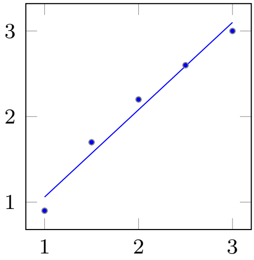
```
>> x = [1.0, 1.5, 2.0, 2.5, 3.0]';
>> y = [0.9, 1.7, 2.2, 2.6, 3.0]';
>> a = polyfit(x,y,1)

a =

     1.0200    0.0400

>> xi = 1:0.1:3;
>> yi = polyval(a,xi);
>> plot(x,y,'o',xi,yi);
fx >>
```



拟合：线性和非线性拟合

Command Window

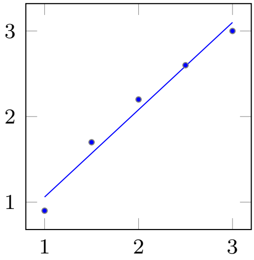
```
>> x = [1.0, 1.5, 2.0, 2.5, 3.0]';
>> y = [0.9, 1.7, 2.2, 2.6, 3.0]';
>> a = polyfit(x,y,1)

a =

      1.0200    0.0400

>> xi = 1:0.1:3;
>> yi = polyval(a,xi);
>> plot(x,y,'o',xi,yi);
>> p = fittype('a*x+b*sin(x)+c');
fx >>
```



拟合：线性和非线性拟合

Command Window

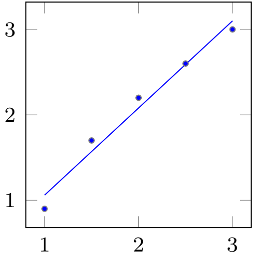
```
>> x = [1.0, 1.5, 2.0, 2.5, 3.0]';
>> y = [0.9, 1.7, 2.2, 2.6, 3.0]';
>> a = polyfit(x,y,1)

a =

      1.0200    0.0400

>> xi = 1:0.1:3;
>> yi = polyval(a,xi);
>> plot(x,y,'o',xi,yi);
>> p = fittype('a*x+b*sin(x)+c');
>> f = fit(x,y,p)

f =
   General model:
   f(x) = a*x+b*sin(x)+c
   Coefficients (with 95% confidence bounds):
   a = 1.249 (0.9856, 1.512)
   b = 0.6357 (0.03185, 1.24)
   c = -0.8611 (-1.773, 0.05094)

fx >>
```

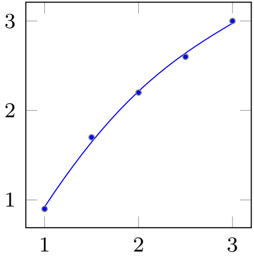


拟合：线性和非线性拟合

Command Window

```
>> x = [1.0, 1.5, 2.0, 2.5, 3.0]';
>> y = [0.9, 1.7, 2.2, 2.6, 3.0]';
>> a = polyfit(x,y,1)

a =

      1.0200    0.0400

>> xi = 1:0.1:3;
>> yi = polyval(a,xi);
>> plot(x,y,'o',xi,yi);
>> p = fittype('a*x+b*sin(x)+c');
>> f = fit(x,y,p)

f =
   General model:
   f(x) = a*x+b*sin(x)+c
   Coefficients (with 95% confidence bounds):
   a = 1.249 (0.9856, 1.512)
   b = 0.6357 (0.03185, 1.24)
   c = -0.8611 (-1.773, 0.05094)

>> plot(f,x,y);
```





拟合：美国人口指数增长模型拟合

1790-1900 年美国人口数

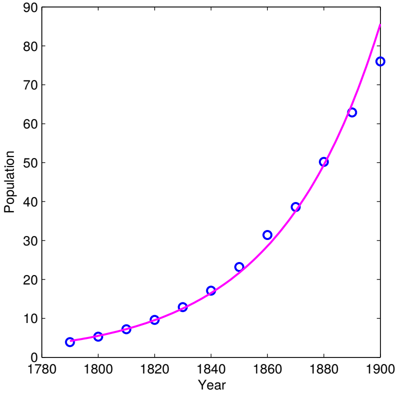
1790	3.9	1840	17.1	1890	62.9
1800	5.3	1850	23.2	1900	76.0
1810	7.2	1860	31.4		
1820	9.6	1870	38.6		
1830	12.9	1880	50.2		

指数增长模型：指数方程转化为线性方程

$$
\begin{aligned}
x(t) &= x_0 \mathrm{e}^{rt} \\
&\Downarrow \\
\ln x(t) &= rt + \ln x_0 \\
&\Downarrow \\
Y &= a_1 t + a_2
\end{aligned}
$$

拟合：美国人口指数增长模型拟合

```
t = 1790:10:1900;
p = [3.9  5.3  7.2  9.6 ...
     12.9 17.1 23.2 31.4 ...
     38.6 50.2 62.9 76.0];

Y = log(p); X = t;
a = polyfit(X,Y,1);
x0 = exp(a(2)); r = a(1);
ti = 1790:1900;
pti= x0*exp(r*ti);
plot(t,p,'o',ti,pti,'m')
xlabel('Year')
ylabel('Population')
```



线性回归

线性回归：regress

$$Y = b_0 1 + b_1 x_1 + b_2 x_2 + \cdots + b_k x_k$$

```
[B,Bint,R,Rint,Stats] = regress(Y,X
```

- B: 回归得到的自变量系数.
- Bint: B 的 95% 的置信区间矩阵
- R: 残差向量
- Rint: 置信区间
- Stats: 统计量, 包含 R 方统计量, F 统计量等

线性回归：牙膏的销售量

问题

- 建立牙膏销售量与价格、广告投入之间的模型
- 预测在不同价格和广告费用下的牙膏销售量

$y = b_0 + b_1 x_1 + b_2 x_2 + b_3 x_2^2 + b_4 x_1 x_2$

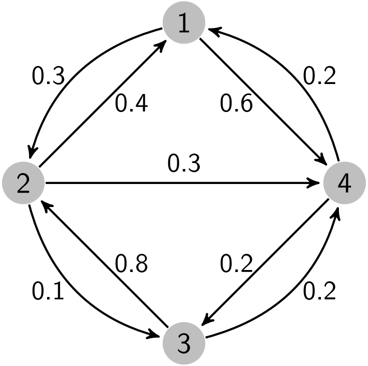
周期	广告费 x_2	价格差（百万）x_1	销售量（百万）y
1	5.50	-0.05	7.38
2	6.75	0.25	8.51
⋮	⋮	⋮	⋮
30	6.80	0.55	9.26

```
x = [ones(30,1), x1, x2, x2.^2, x1.*x2]
[b, bint, r, rint, stats] = regress(y,x)
```

提要

1. 评价和预测
   - 层次分析
   - 拟合回归
2. 图论与网络
   - 图论算法的实现：最短路径
   - 网络算法的实现：pagerank
3. 模拟退火
   - 算法启源和思想
   - 应用举例

图论的定义



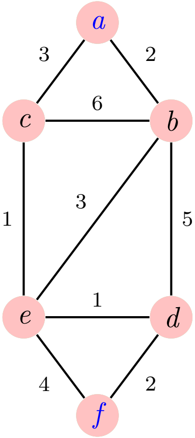
- 图论（Graph theory）以图为研究对象，研究顶点和边组成的图形的数学理论和方法.
- 图论中的图是由若干给定的顶点及连接两顶点的边所构成的图形.
- 图论中的图通常用来描述某些事物之间的某种特定关系，用顶点代表事物，用边表示相应两个事物间的关系.

图论工具介绍：主要命令

图论工具箱的相关命令

函数名	功能
graphallshortestpaths	求图中所有顶点对之间的最短距离
graphconnredcomp	找无（有）向图的（强/弱）连通分支
graphisreddag	测试有向图是否含有圈
graphisomorphism	确定一个图是否有生成树
graphmaxflow	计算有向图的最大流
graphminspantree	在图中找最小生成树
graphpred2path	把前驱顶点序列变成路径的顶点序列
graphshortestpath	求指定一对顶点间的最短距离和路径
graphtopoorder	执行有向无圈图的拓扑排序
graphtraverse	求从一顶点出发，所能遍历图中的顶点

图论工具介绍：普通矩阵 $\rightleftharpoons$ 稀疏矩阵



满矩阵和稀疏矩阵 (full$\rightleftharpoons$sparse)

$$\begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 2 & 0 & 0 & 0 & 0 & 0 \\ 3 & 6 & 0 & 0 & 0 & 0 \\ 0 & 5 & 0 & 0 & 0 & 0 \\ 0 & 3 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 2 & 4 & 0 \end{bmatrix} \rightleftharpoons \begin{matrix} (2,1) & 2 \\ (3,1) & 3 \\ (3,2) & 6 \\ (4,2) & 5 \\ (5,2) & 3 \\ (5,3) & 1 \\ (5,4) & 1 \\ (6,4) & 2 \\ (6,5) & 4 \end{matrix}$$

图论工具介绍：实现最短路径算法

graphshortestpath 函数用法

```
[a,b,c,d,e,f] = deal(1,2,3,4,5,6);
%      a  b  c  d  e  f
w = [ 0  2  3  0  0  0    % a
      2  0  6  5  3  0    % b
      3  6  0  0  1  0    % c
      0  5  0  0  1  2    % d
      0  3  1  1  0  4    % e
      0  0  0  2  4  0]; % f

W = sparse(w);
[dist, path, pred] = graphshortestpath(W, a, f)
```

图论工具介绍：实现最短路径算法

graphshortestpath 函数用法

```
[a,b,c,d,e,f] = deal(1,2,3,4,5,6);
%      a  b  c  d  e  f
w = [  0  2  3  0  0  0   % a
       2  0  6  5  3  0   % b
       3  6  0  0  1  0   % c
       0  5  0  0  1  2   % d
       0  3  1  1  0  4   % e
       0  0  0  2  4  0]; % f

W = sparse(w);
[dist, path, pred] = graphshortestpath(W, a, f)
```

图论工具介绍：实现最短路径算法

graphshortestpath 函数用法

```
[a,b,c,d,e,f] = deal(1,2,3,4,5,6);
%        a   b   c   d   e   f
w = [  0   2   3   0   0   0    % a
       2   0   6   5   3   0    % b
       3   6   0   0   1   0    % c
       0   5   0   0   1   2    % d
       0   3   1   1   0   4    % e
       0   0   0   2   4   0];  % f

W = sparse(w);
[dist, path, pred] = graphshortestpath(W, a, f)
```

图论工具介绍：实现最短路径算法

graphshortestpath 函数用法

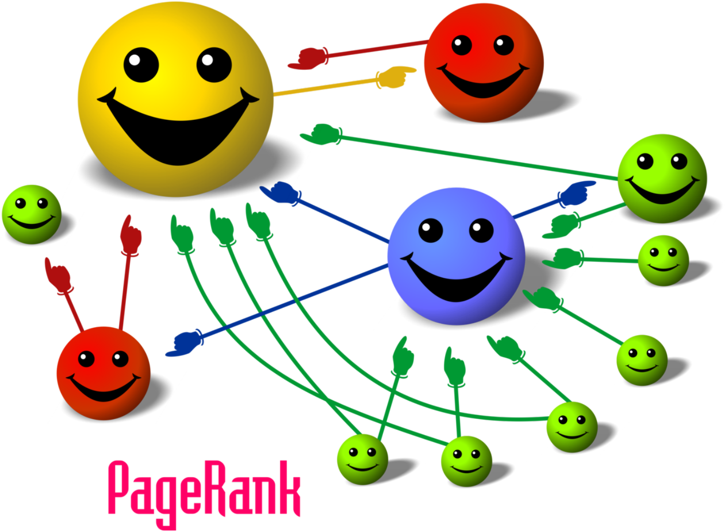
```
[a,b,c,d,e,f] = deal(1,2,3,4,5,6);
%       a   b   c   d   e   f
w = [ 0   2   3   0   0   0    % a
      2   0   6   5   3   0    % b
      3   6   0   0   1   0    % c
      0   5   0   0   1   2    % d
      0   3   1   1   0   4    % e
      0   0   0   2   4   0]; % f

W = sparse(w);
[dist, path, pred] = graphshortestpath(W, a, f)
```

图论工具介绍：实现最短路径算法

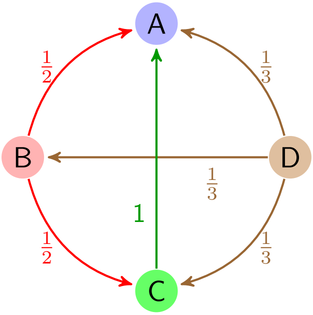
graphshortestpath 函数用法

```
[a,b,c,d,e,f] = deal(1,2,3,4,5,6);
%       a   b   c   d   e   f
w = [   0   2   3   0   0   0    % a
        2   0   6   5   3   0    % b
        3   6   0   0   1   0    % c
        0   5   0   0   1   2    % d
        0   3   1   1   0   4    % e
        0   0   0   2   4   0];  % f

W = sparse(w);
[dist, path, pred] = graphshortestpath(W, a, f)
```

网页排序算法：pagerank



网页排序算法：pagerank



$$R(A) = \frac{R(B)}{2} + \frac{R(C)}{1} + \frac{R(D)}{3}$$

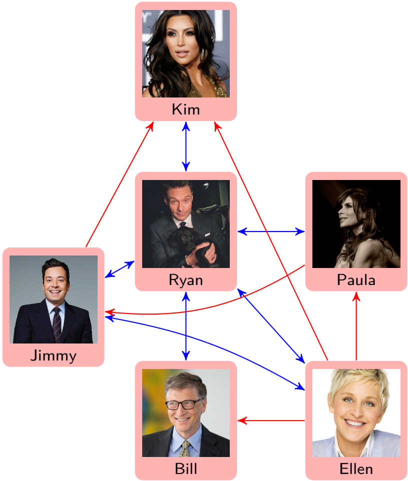
指标形式

$$R(p_i) = \frac{1-d}{N} + d \sum_{p_j \in M(p_i)} \frac{R(p_j)}{L(p_j)}$$

矩阵形式

$$\mathbf{R} = \begin{bmatrix} \frac{1-d}{N} \\ \vdots \\ \frac{1-d}{N} \end{bmatrix} + d \begin{bmatrix} l_{1,1} & \cdots & l_{1,n} \\ \vdots & \ddots & \vdots \\ l_{n,1} & \cdots & l_{n,n} \end{bmatrix} \mathbf{R}$$

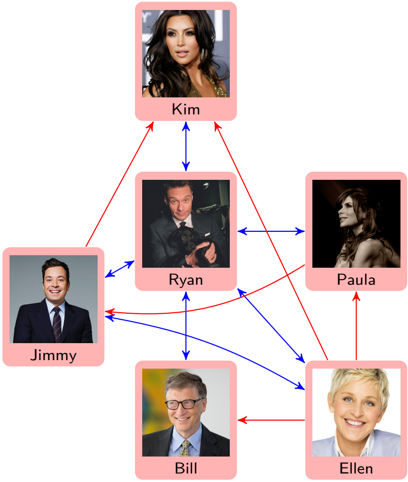
pagerank 算法的实现



celebrity.m

```
d = 0.85;
n = 6;

C = (1-d)/n*ones(n,1);

L=[0 1/5  0   0   0  1/5 %bill
   0  0  1/3  0   0  1/5 %ellen
   0 1/5  0   0  1/2 1/5 %jimmy
   0 1/5 1/3  0   0  1/5 %kim
   0 1/5  0   0   0  1/5 %paula
   1 1/5 1/3  1  1/2  0];%ryan

I = eye(n);

R = (I - d*L)\C % R = C+d*L*R
```

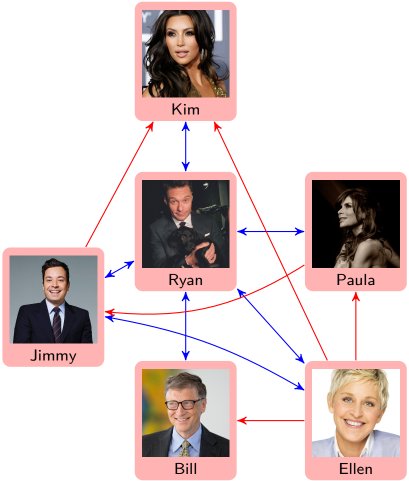
pagerank 算法的实现



celebrity.m

```
d = 0.85;
n = 6;

C = (1-d)/n*ones(n,1);

L=[0 1/5   0   0   0   1/5 %bill
     0   0  1/3  0   0   1/5 %ellen
     0 1/5   0   0  1/2  1/5 %jimmy
     0 1/5  1/3  0   0   1/5 %kim
     0 1/5   0   0   0   1/5 %paula
     1 1/5  1/3  1  1/2   0];%ryan

I = eye(n);

R = (I - d*L)\C % R = C+d*L*R
```

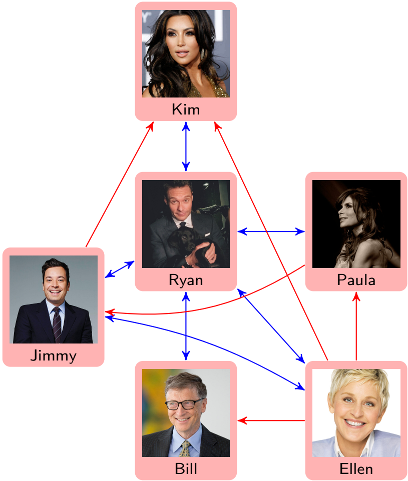
pagerank 算法的实现



celebrity.m

```
d = 0.85;
n = 6;

C = (1-d)/n*ones(n,1);

L=[0 1/5  0   0  0   1/5 %bill
    0  0  1/3 0  0   1/5 %ellen
    0 1/5  0  0 1/2  1/5 %jimmy
    0 1/5 1/3 0  0   1/5 %kim
    0 1/5  0  0  0   1/5 %paula
    1 1/5 1/3 1 1/2   0];%ryan

I = eye(n);

R = (I - d*L)\C % R = C+d*L*R
```

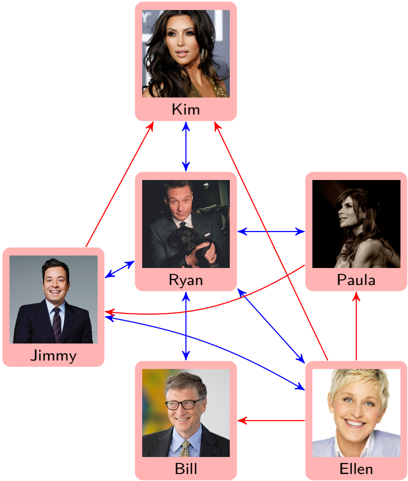
pagerank 算法的实现



celebrity.m

```
d = 0.85;
n = 6;

C = (1-d)/n*ones(n,1);

L=[0 1/5   0   0   0   1/5 %bill
   0   0 1/3   0   0   1/5 %ellen
   0 1/5   0   0 1/2   1/5 %jimmy
   0 1/5 1/3   0   0   1/5 %kim
   0 1/5   0   0   0   1/5 %paula
   1 1/5 1/3   1 1/2    0];%ryan

I = eye(n);

R = (I - d*L)\C % R = C+d*L*R
```

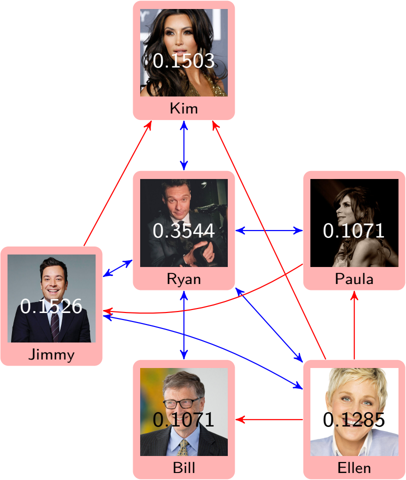
pagerank 算法的实现



celebrity.m

```
d = 0.85;
n = 6;

C = (1-d)/n*ones(n,1);

L=[0 1/5   0   0   0   1/5 %bill
     0   0  1/3  0   0   1/5 %ellen
     0 1/5   0   0  1/2  1/5 %jimmy
     0 1/5  1/3  0   0   1/5 %kim
     0 1/5   0   0   0   1/5 %paula
     1 1/5  1/3  1  1/2   0];%ryan

I = eye(n);

R = (I - d*L)\C % R = C+d*L*R
```

pagerank 算法的实现



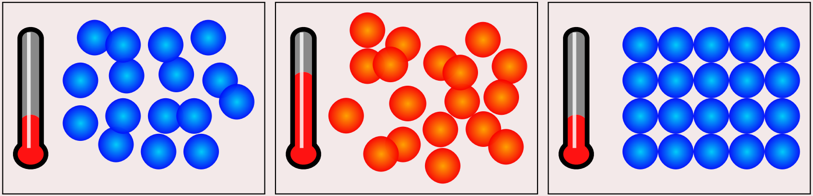
celebrity.m

```
d = 0.85;
n = 6;

C = (1-d)/n*ones(n,1);

L=[0 1/5   0   0   0   1/5 %bill
   0   0 1/3   0   0   1/5 %ellen
   0 1/5   0   0 1/2   1/5 %jimmy
   0 1/5 1/3   0   0   1/5 %kim
   0 1/5   0   0   0   1/5 %paula
   1 1/5 1/3   1 1/2    0];%ryan

I = eye(n);

R = (I - d*L)\C % R = C+d*L*R
```

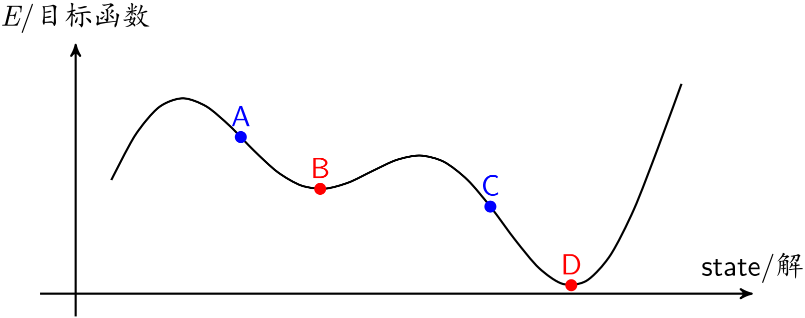
提要

物理退火

物理退火过程



模拟退火算法与 Metropolis 准则



$$P_{\text{accept}}(E_{i+1} \leq E_i) = 1, \ \ P_{\text{accept}}(E_{i+1} > E_i) = \mathrm{e}^{-(E_{i+1}-E_i)/KT}$$

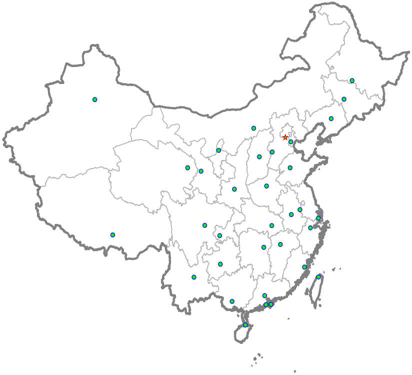
模拟退火算法基本思想

模拟退火算法伪代码

```
construct initial solution x_0, and x^current = x_0
set initial temperature T = T_0
while continuing criterion do
    for i = 1 to T_L do
        generate randomly a neighbouring solution x' ∈ N(x^current)
        compute change of cost ΔC = C(x') − C(x^current)
        if ΔC ≤ 0 or random(0,1) < exp(−ΔC/(kT)) then
            x^current = x' {accept new state}
        end if
    end for
    set new temperature T = decrease(T) {decrease temperature}
end while
return solution corresponding to the minimum cost function
```

TSP 问题的模拟退火求解：问题

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. 用模拟退火算法求最短路径.



- 如何设置初始解 S_0?

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

- 如何产生邻解 S'?

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

- 如何定义 COST 函数?

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

- 如何设置温度，降温?

  $T_0 = 1000, T_{\tau+d\tau} = \alpha T_\tau$

TSP 问题的模拟退火求解: 程序

主程序 MatLab 代码

```
route = randperm(numberofcities); %路径格式:[1,2,...,n]
temperature = 1000; cooling_rate = 0.95;   %初始化温度
Titerations = 1;                          %用来控制降温的循环
previous_distance = totaldistance(route);  %计算路径总长
while temperature > 1.0                   %循环继续条件
   temp_route = perturb(route, 'reverse'); %扰动产生邻解
   current_distance = totaldistance(temp_route);%路长
   diff = current_distance - previous_distance;
   if (diff<0)||(rand < exp(-diff/(temperature)))
      route = temp_route;                %接受当前解
      previous_distance = current_distance;
      Titerations = Titerations + 1;
   end
   if Titerations >= 10                  %每10步降温(等温步数为10)
      temperature = cooling_rate*temperature;
      Titerations = 0;
   end
end
```

TSP 问题的模拟退火求解：程序

主程序 MatLab 代码

```
route = randperm(numberofcities); %路径格式:[1,2,...,n]
temperature = 1000; cooling_rate = 0.95;   %初始化温度
Titerations = 1;                          %用来控制降温的循环
previous_distance = totaldistance(route);  %计算路径总长
while temperature > 1.0                   %循环继续条件
   temp_route = perturb(route, 'reverse'); %扰动产生邻解
   current_distance = totaldistance(temp_route);%路长
   diff = current_distance - previous_distance;
   if (diff<0)||(rand < exp(-diff/(temperature)))
      route = temp_route;                %接受当前解
      previous_distance = current_distance;
      Titerations = Titerations + 1;
   end
   if Titerations >= 10                  %每10步降温(等温步数为10)
      temperature = cooling_rate*temperature;
      Titerations = 0;
   end
end
```

TSP 问题的模拟退火求解：程序

主程序 MatLab 代码

```
route = randperm(numberofcities); %路径格式:[1,2,...,n]
temperature = 1000; cooling_rate = 0.95;   %初始化温度
Titerations = 1;                          %用来控制降温的循环
previous_distance = totaldistance(route);  %计算路径总长
while temperature > 1.0                   %循环继续条件
   temp_route = perturb(route, 'reverse'); %扰动产生邻解
   current_distance = totaldistance(temp_route);%路长
   diff = current_distance - previous_distance;
   if (diff<0)||(rand < exp(-diff/(temperature)))
      route = temp_route;                %接受当前解
      previous_distance = current_distance;
      Titerations = Titerations + 1;
   end
   if Titerations >= 10                  %每10步降温(等温步数为10)
      temperature = cooling_rate*temperature;
      Titerations = 0;
   end
end
```

TSP 问题的模拟退火求解: 程序

主程序 MatLab 代码

```
route = randperm(numberofcities); %路径格式:[1,2,...,n]
temperature = 1000; cooling_rate = 0.95;   %初始化温度
Titerations = 1;                          %用来控制降温的循环
previous_distance = totaldistance(route);  %计算路径总长
while temperature > 1.0                   %循环继续条件
   temp_route = perturb(route, 'reverse'); %扰动产生邻解
   current_distance = totaldistance(temp_route);%路长
   diff = current_distance - previous_distance;
   if (diff<0)||(rand < exp(-diff/(temperature)))
      route = temp_route;               %接受当前解
      previous_distance = current_distance;
      Titerations = Titerations + 1;
   end
   if Titerations >= 10                 %每10步降温(等温步数为10)
      temperature = cooling_rate*temperature;
      Titerations = 0;
   end
end
```

TSP 问题的模拟退火求解：程序

主程序 MatLab 代码

```matlab
route = randperm(numberofcities); %路径格式:[1,2,...,n]
temperature = 1000; cooling_rate = 0.95;   %初始化温度
Titerations = 1;                         %用来控制降温的循环
previous_distance = totaldistance(route);  %计算路径总长
while temperature > 1.0                  %循环继续条件
   temp_route = perturb(route, 'reverse'); %扰动产生邻解
   current_distance = totaldistance(temp_route);%路长
   diff = current_distance - previous_distance;
   if (diff<0)||(rand < exp(-diff/(temperature)))
      route = temp_route;            %接受当前解
      previous_distance = current_distance;
      Titerations = Titerations + 1;
   end
   if Titerations >= 10             %每10步降温(等温步数为10)
      temperature = cooling_rate*temperature;
      Titerations = 0;
   end
end
```

TSP 问题的模拟退火求解：程序

主程序 MatLab 代码

```
route = randperm(numberofcities); %路径格式:[1,2,...,n]
temperature = 1000; cooling_rate = 0.95;   %初始化温度
Titerations = 1;                        %用来控制降温的循环
previous_distance = totaldistance(route);  %计算路径总长
while temperature > 1.0                 %循环继续条件
   temp_route = perturb(route, 'reverse'); %扰动产生邻解
   current_distance = totaldistance(temp_route);%路长
   diff = current_distance - previous_distance;
   if (diff<0)||(rand < exp(-diff/(temperature)))
      route = temp_route;          %接受当前解
      previous_distance = current_distance;
      Titerations = Titerations + 1;
   end
   if Titerations >= 10            %每10步降温(等温步数为10)
      temperature = cooling_rate*temperature;
      Titerations = 0;
   end
end
```

TSP 问题的模拟退火求解: 程序

主程序 MatLab 代码

```
route = randperm(numberofcities); %路径格式:[1,2,...,n]
temperature = 1000; cooling_rate = 0.95;   %初始化温度
Titerations = 1;                           %用来控制降温的循环
previous_distance = totaldistance(route);  %计算路径总长
while temperature > 1.0                    %循环继续条件
   temp_route = perturb(route, 'reverse'); %扰动产生邻解
   current_distance = totaldistance(temp_route);%路长
   diff = current_distance - previous_distance;
   if (diff<0)||(rand < exp(-diff/(temperature)))
      route = temp_route;                 %接受当前解
      previous_distance = current_distance;
      Titerations = Titerations + 1;
   end
   if Titerations >= 10                   %每10步降温(等温步数为10)
      temperature = cooling_rate*temperature;
      Titerations = 0;
   end
end
```

TSP 问题的模拟退火求解: 程序

主程序 MatLab 代码

```
route = randperm(numberofcities); %路径格式:[1,2,...,n]
temperature = 1000; cooling_rate = 0.95;   %初始化温度
Titerations = 1;                          %用来控制降温的循环
previous_distance = totaldistance(route);  %计算路径总长
while temperature > 1.0                   %循环继续条件
   temp_route = perturb(route, 'reverse'); %扰动产生邻解
   current_distance = totaldistance(temp_route);%路长
   diff = current_distance - previous_distance;
   if (diff<0)||(rand < exp(-diff/(temperature)))
      route = temp_route;          %接受当前解
      previous_distance = current_distance;
      Titerations = Titerations + 1;
   end
   if Titerations >= 10            %每10步降温(等温步数为10)
      temperature = cooling_rate*temperature;
      Titerations = 0;
   end
end
```

TSP 问题的模拟退火求解：程序

主程序 MatLab 代码

```
route = randperm(numberofcities); %路径格式:[1,2,...,n]
temperature = 1000; cooling_rate = 0.95;   %初始化温度
Titerations = 1;                          %用来控制降温的循环
previous_distance = totaldistance(route);  %计算路径总长
while temperature > 1.0                   %循环继续条件
   temp_route = perturb(route, 'reverse'); %扰动产生邻解
   current_distance = totaldistance(temp_route);%路长
   diff = current_distance - previous_distance;
   if (diff<0)||(rand < exp(-diff/(temperature)))
      route = temp_route;           %接受当前解
      previous_distance = current_distance;
      Titerations = Titerations + 1;
   end
   if Titerations >= 10             %每10步降温(等温步数为10)
      temperature = cooling_rate*temperature;
      Titerations = 0;
   end
end
```

TSP 问题的模拟退火求解: 程序

主程序 MatLab 代码

```
route = randperm(numberofcities); %路径格式:[1,2,...,n]
temperature = 1000; cooling_rate = 0.95;   %初始化温度
Titerations = 1;                          %用来控制降温的循环
previous_distance = totaldistance(route);  %计算路径总长
while temperature > 1.0                   %循环继续条件
   temp_route = perturb(route, 'reverse'); %扰动产生邻解
   current_distance = totaldistance(temp_route);%路长
   diff = current_distance - previous_distance;
   if (diff<0)||(rand < exp(-diff/(temperature)))
      route = temp_route;          %接受当前解
      previous_distance = current_distance;
      Titerations = Titerations + 1;
   end
   if Titerations >= 10           %每10步降温(等温步数为10)
      temperature = cooling_rate*temperature;
      Titerations = 0;
   end
end
```

TSP 问题的模拟退火求解: 程序

主程序 MatLab 代码

```
route = randperm(numberofcities); %路径格式:[1,2,...,n]
temperature = 1000; cooling_rate = 0.95;   %初始化温度
Titerations = 1;                          %用来控制降温的循环
previous_distance = totaldistance(route);  %计算路径总长
while temperature > 1.0                   %循环继续条件
   temp_route = perturb(route, 'reverse'); %扰动产生邻解
   current_distance = totaldistance(temp_route);%路长
   diff = current_distance - previous_distance;
   if (diff<0)||(rand < exp(-diff/(temperature)))
      route = temp_route;           %接受当前解
      previous_distance = current_distance;
      Titerations = Titerations + 1;
   end
   if Titerations >= 10             %每10步降温(等温步数为10)
      temperature = cooling_rate*temperature;
      Titerations = 0;
   end
end
```

TSP 问题的模拟退火求解: 程序

主程序 MatLab 代码

```
route = randperm(numberofcities); %路径格式:[1,2,...,n]
temperature = 1000; cooling_rate = 0.95;   %初始化温度
Titerations = 1;                           %用来控制降温的循环
previous_distance = totaldistance(route);  %计算路径总长
while temperature > 1.0                    %循环继续条件
   temp_route = perturb(route, 'reverse'); %扰动产生邻解
   current_distance = totaldistance(temp_route);%路长
   diff = current_distance - previous_distance;
   if (diff<0)||(rand < exp(-diff/(temperature)))
      route = temp_route;                 %接受当前解
      previous_distance = current_distance;
      Titerations = Titerations + 1;
   end
   if Titerations >= 10                   %每10步降温(等温步数为10)
      temperature = cooling_rate*temperature;
      Titerations = 0;
   end
end
```

TSP 问题的模拟退火求解: 程序

距离矩阵函数 distancematrix

```matlab
function dis = distancematrix(city)
numberofcities = length(city);
R = 6378.137; %地球半径, 用于求两个城市的球面距离
for i = 1:numberofcities
   for j = i+1:numberofcities
      dis(i,j) = distance(city(i).lat, city(i).long, ...
                          city(j).lat, city(j).long, R);
      dis(j,i) = dis(i,j);
   end
end
```

路径距离计算函数 totaldistance

```matlab
function d = totaldistance(dis, route)
d = dis(route(end),route(1));
for k = 1:length(route)-1
    i = route(k); j = route(k+1);
    d = d + dis(i,j);
end
```

TSP 问题的模拟退火求解: 程序

距离矩阵函数 distancematrix

```
function dis = distancematrix(city)
numberofcities = length(city);
R = 6378.137; %地球半径, 用于求两个城市的球面距离
for i = 1:numberofcities
   for j = i+1:numberofcities
      dis(i,j) = distance(city(i).lat, city(i).long, ...
                           city(j).lat, city(j).long, R);
      dis(j,i) = dis(i,j);
   end
end
```

路径距离计算函数 totaldistance

```
function d = totaldistance(dis, route)
d = dis(route(end),route(1));
for k = 1:length(route)-1
    i = route(k); j = route(k+1);
    d = d + dis(i,j);
end
```

TSP 问题的模拟退火求解: 程序

距离矩阵函数 distancematrix

```
function dis = distancematrix(city)
numberofcities = length(city);
R = 6378.137; %地球半径, 用于求两个城市的球面距离
for i = 1:numberofcities
   for j = i+1:numberofcities
      dis(i,j) = distance(city(i).lat, city(i).long, ...
                          city(j).lat, city(j).long, R);
      dis(j,i) = dis(i,j);
   end
end
```

路径距离计算函数 totaldistance

```
function d = totaldistance(dis, route)
d = dis(route(end),route(1));
for k = 1:length(route)-1
    i = route(k); j = route(k+1);
    d = d + dis(i,j);
end
```

TSP 问题的模拟退火求解：程序

距离矩阵函数 distancematrix

```
function dis = distancematrix(city)
numberofcities = length(city);
R = 6378.137; %地球半径，用于求两个城市的球面距离
for i = 1:numberofcities
   for j = i+1:numberofcities
      dis(i,j) = distance(city(i).lat, city(i).long, ...
                           city(j).lat, city(j).long, R);
      dis(j,i) = dis(i,j);
   end
end
```

路径距离计算函数 totaldistance

```
function d = totaldistance(dis, route)
d = dis(route(end),route(1));
for k = 1:length(route)-1
    i = route(k); j = route(k+1);
    d = d + dis(i,j);
end
```

TSP 问题的模拟退火求解：程序

距离矩阵函数 distancematrix

```
function dis = distancematrix(city)
numberofcities = length(city);
R = 6378.137; %地球半径，用于求两个城市的球面距离
for i = 1:numberofcities
   for j = i+1:numberofcities
      dis(i,j) = distance(city(i).lat, city(i).long, ...
                           city(j).lat, city(j).long, R);
      dis(j,i) = dis(i,j);
   end
end
```

路径距离计算函数 totaldistance

```
function d = totaldistance(dis, route)
d = dis(route(end),route(1));
for k = 1:length(route)-1
    i = route(k); j = route(k+1);
    d = d + dis(i,j);
end
```

TSP 问题的模拟退火求解：程序

距离矩阵函数 distancematrix

```
function dis = distancematrix(city)
numberofcities = length(city);
R = 6378.137; %地球半径，用于求两个城市的球面距离
for i = 1:numberofcities
   for j = i+1:numberofcities
      dis(i,j) = distance(city(i).lat, city(i).long, ...
                          city(j).lat, city(j).long, R);
      dis(j,i) = dis(i,j);
   end
end
```

路径距离计算函数 totaldistance

```
function d = totaldistance(dis, route)
d = dis(route(end),route(1));
for k = 1:length(route)-1
    i = route(k); j = route(k+1);
    d = d + dis(i,j);
end
```

TSP 问题的模拟退火求解: 程序

距离矩阵函数 distancematrix

```
function dis = distancematrix(city)
numberofcities = length(city);
R = 6378.137; %地球半径, 用于求两个城市的球面距离
for i = 1:numberofcities
   for j = i+1:numberofcities
      dis(i,j) = distance(city(i).lat, city(i).long, ...
                          city(j).lat, city(j).long, R);
      dis(j,i) = dis(i,j);
   end
end
```

路径距离计算函数 totaldistance

```
function d = totaldistance(dis, route)
d = dis(route(end),route(1));
for k = 1:length(route)-1
    i = route(k); j = route(k+1);
    d = d + dis(i,j);
end
```

TSP 问题的模拟退火求解: 程序

距离矩阵函数 distancematrix

```
function dis = distancematrix(city)
numberofcities = length(city);
R = 6378.137; %地球半径, 用于求两个城市的球面距离
for i = 1:numberofcities
   for j = i+1:numberofcities
      dis(i,j) = distance(city(i).lat, city(i).long, ...
                          city(j).lat, city(j).long, R);
      dis(j,i) = dis(i,j);
   end
end
```

路径距离计算函数 totaldistance

```
function d = totaldistance(dis, route)
d = dis(route(end),route(1));
for k = 1:length(route)-1
    i = route(k); j = route(k+1);
    d = d + dis(i,j);
end
```

TSP 问题的模拟退火求解：程序

距离矩阵函数 distancematrix

```
function dis = distancematrix(city)
numberofcities = length(city);
R = 6378.137; %地球半径，用于求两个城市的球面距离
for i = 1:numberofcities
   for j = i+1:numberofcities
      dis(i,j) = distance(city(i).lat, city(i).long, ...
                          city(j).lat, city(j).long, R);
      dis(j,i) = dis(i,j);
   end
end
```

路径距离计算函数 totaldistance

```
function d = totaldistance(dis, route)
d = dis(route(end),route(1));
for k = 1:length(route)-1
    i = route(k); j = route(k+1);
    d = d + dis(i,j);
end
```

TSP 问题的模拟退火求解：程序

距离矩阵函数 distancematrix

```
function dis = distancematrix(city)
numberofcities = length(city);
R = 6378.137; %地球半径，用于求两个城市的球面距离
for i = 1:numberofcities
   for j = i+1:numberofcities
      dis(i,j) = distance(city(i).lat, city(i).long, ...
                          city(j).lat, city(j).long, R);
      dis(j,i) = dis(i,j);
   end
end
```

路径距离计算函数 totaldistance

```
function d = totaldistance(dis, route)
d = dis(route(end),route(1));
for k = 1:length(route)-1
    i = route(k); j = route(k+1);
    d = d + dis(i,j);
end
```

TSP 问题的模拟退火求解: 程序

距离矩阵函数 distancematrix

```
function dis = distancematrix(city)
numberofcities = length(city);
R = 6378.137; %地球半径, 用于求两个城市的球面距离
for i = 1:numberofcities
   for j = i+1:numberofcities
      dis(i,j) = distance(city(i).lat, city(i).long, ...
                          city(j).lat, city(j).long, R);
      dis(j,i) = dis(i,j);
   end
end
```

路径距离计算函数 totaldistance

```
function d = totaldistance(dis, route)
d = dis(route(end),route(1));
for k = 1:length(route)-1
    i = route(k); j = route(k+1);
    d = d + dis(i,j);
end
```

TSP 问题的模拟退火求解: 程序

距离矩阵函数 distancematrix

```
function dis = distancematrix(city)
numberofcities = length(city);
R = 6378.137; %地球半径, 用于求两个城市的球面距离
for i = 1:numberofcities
   for j = i+1:numberofcities
      dis(i,j) = distance(city(i).lat, city(i).long, ...
                          city(j).lat, city(j).long, R);
      dis(j,i) = dis(i,j);
   end
end
```

路径距离计算函数 totaldistance

```
function d = totaldistance(dis, route)
d = dis(route(end),route(1));
for k = 1:length(route)-1
    i = route(k); j = route(k+1);
    d = d + dis(i,j);
end
```

TSP 问题的模拟退火求解: 程序

距离矩阵函数 distancematrix

```
function dis = distancematrix(city)
numberofcities = length(city);
R = 6378.137; %地球半径，用于求两个城市的球面距离
for i = 1:numberofcities
   for j = i+1:numberofcities
      dis(i,j) = distance(city(i).lat, city(i).long, ...
                          city(j).lat, city(j).long, R);
      dis(j,i) = dis(i,j);
   end
end
```

路径距离计算函数 totaldistance

```
function d = totaldistance(dis, route)
d = dis(route(end),route(1));
for k = 1:length(route)-1
    i = route(k); j = route(k+1);
    d = d + dis(i,j);
end
```

TSP 问题的模拟退火求解: 程序

产生邻解的函数 perturb

```
function route = perturb(route_old,method)
route = route_old;
numbercities = length(route);
city1 = ceil(numbercities*rand);     %  [1, 2, ..., n-1, n]
city2 = ceil(numbercities*rand);     %  1<=city1, city2<=n
switch method
    case 'reverse'           %[1 2 3 4 5 6] -> [1 5 4 3 2 6]
        cmin = min(city1,city2);
        cmax = max(city1,city2);
        route(cmin:cmax) = route(cmax:-1:cmin);
    case 'swap'              %[1 2 3 4 5 6] -> [1 5 3 4 2 6]
        route([city1, city2]) = route([city2, city1]);
end
```

TSP 问题的模拟退火求解：程序

产生邻解的函数 perturb

```
function route = perturb(route_old,method)
route = route_old;
numbercities = length(route);
city1 = ceil(numbercities*rand);     %  [1, 2, ..., n-1, n]
city2 = ceil(numbercities*rand);     %  1<=city1, city2<=n
switch method
    case 'reverse'          %[1 2 3 4 5 6] -> [1 5 4 3 2 6]
        cmin = min(city1,city2);
        cmax = max(city1,city2);
        route(cmin:cmax) = route(cmax:-1:cmin);
    case 'swap'             %[1 2 3 4 5 6] -> [1 5 3 4 2 6]
        route([city1, city2]) = route([city2, city1]);
end
```

TSP 问题的模拟退火求解: 程序

产生邻解的函数 perturb

```
function route = perturb(route_old,method)
route = route_old;
numbercities = length(route);
city1 = ceil(numbercities*rand);     %  [1, 2, ..., n-1, n]
city2 = ceil(numbercities*rand);     %  1<=city1, city2<=n
switch method
    case 'reverse'          %[1 2 3 4 5 6] -> [1 5 4 3 2 6]
        cmin = min(city1,city2);
        cmax = max(city1,city2);
        route(cmin:cmax) = route(cmax:-1:cmin);
    case 'swap'             %[1 2 3 4 5 6] -> [1 5 3 4 2 6]
        route([city1, city2]) = route([city2, city1]);
end
```

TSP 问题的模拟退火求解：程序

产生邻解的函数 perturb

```
function route = perturb(route_old,method)
route = route_old;
numbercities = length(route);
city1 = ceil(numbercities*rand);     %  [1, 2, ..., n-1, n]
city2 = ceil(numbercities*rand);     %  1<=city1, city2<=n
switch method
    case 'reverse'           %[1 2 3 4 5 6] -> [1 5 4 3 2 6]
        cmin = min(city1,city2);
        cmax = max(city1,city2);
        route(cmin:cmax) = route(cmax:-1:cmin);
    case 'swap'              %[1 2 3 4 5 6] -> [1 5 3 4 2 6]
        route([city1, city2]) = route([city2, city1]);
end
```

TSP 问题的模拟退火求解：程序

产生邻解的函数 perturb

```
function route = perturb(route_old,method)
route = route_old;
numbercities = length(route);
city1 = ceil(numbercities*rand);     %  [1, 2, ..., n-1, n]
city2 = ceil(numbercities*rand);     %  1<=city1, city2<=n
switch method
    case 'reverse'          %[1 2 3 4 5 6] -> [1 5 4 3 2 6]
        cmin = min(city1,city2);
        cmax = max(city1,city2);
        route(cmin:cmax) = route(cmax:-1:cmin);
    case 'swap'             %[1 2 3 4 5 6] -> [1 5 3 4 2 6]
        route([city1, city2]) = route([city2, city1]);
end
```

TSP 问题的模拟退火求解: 程序

产生邻解的函数 perturb

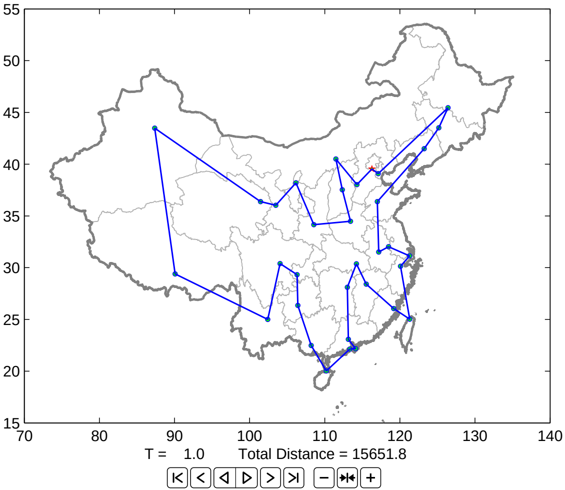
```
function route = perturb(route_old,method)
route = route_old;
numbercities = length(route);
city1 = ceil(numbercities*rand);    %  [1, 2, ..., n-1, n]
city2 = ceil(numbercities*rand);    %  1<=city1, city2<=n
switch method
    case 'reverse'          %[1 2 3 4 5 6] -> [1 5 4 3 2 6]
        cmin = min(city1,city2);
        cmax = max(city1,city2);
        route(cmin:cmax) = route(cmax:-1:cmin);
    case 'swap'             %[1 2 3 4 5 6] -> [1 5 3 4 2 6]
        route([city1, city2]) = route([city2, city1]);
end
```

TSP 问题的模拟退火求解：程序

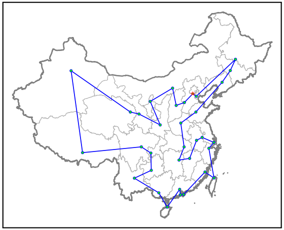
产生邻解的函数 perturb

```
function route = perturb(route_old,method)
route = route_old;
numbercities = length(route);
city1 = ceil(numbercities*rand);     %  [1, 2, ..., n-1, n]
city2 = ceil(numbercities*rand);     %  1<=city1, city2<=n
switch method
    case 'reverse'          %[1 2 3 4 5 6] -> [1 5 4 3 2 6]
        cmin = min(city1,city2);
        cmax = max(city1,city2);
        route(cmin:cmax) = route(cmax:-1:cmin);
    case 'swap'             %[1 2 3 4 5 6] -> [1 5 3 4 2 6]
        route([city1, city2]) = route([city2, city1]);
end
```

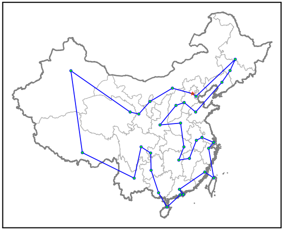
TSP 问题的模拟退火求解：结果



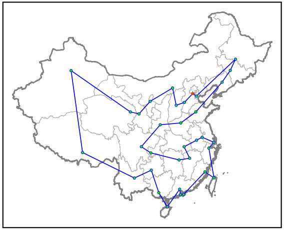
TSP 问题的模拟退火求解：结果



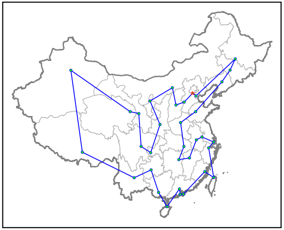
Total Distance = 15757.3



Total Distance = 15775.6



Total Distance = 15691.0



Total Distance = 15905.9

Thank You!!!