曲阜師範大學

学生实验报告

学院 计算机学院

课程 人工智能实训

姓名 彭清元

学号 20204162

年级 2020 级

专业 计算机科学与技术

曲阜师范大学实验报告

学院 计算机学院 年级、专业 20 级计算机科学与技术二班 学号 2020416240

姓名 彭清元 课程名称 人工智能 上课时间 第 1 节 — 第 11 节

实验日期 2023 年 5 月 27 日 星期 六 教师签名 雷玉霞 成绩

实验四 模拟退火算法解决 TSP 问题

1. 【实验目的】

理解并熟悉掌握模拟退火算法算法。

运用模拟退火算法解决 TSP 问题。

2. 【实验原理】

旅行商问题：已知 N 个城市之间的相互距离，现有一个商人必须遍访这 N 个城市，并且每个城市只能访问一次，最后必须返回出发城市。如何安排他对这些城市的访问次序，使其旅行路线总长度最短。

模拟退火算法：模拟退火算法的核心思想就是以一定概率接受比当前解更差的解，然后用这个更差的解继续搜索。先从一个较高的初始温度出发，逐渐降低温度，直到温度降低到满足热平衡条件为止。在每个温度下，进行 n 轮搜索，每轮搜索时对旧解添加随机扰动生成新解，并按一定规则接受新解。模拟退火算法从某一较高初温出发，伴随温度参数的不断下降,结合概率突跳特性在解空间中随机寻找目标函数的全局最优解，即在局部最优解能概率性地跳出并最终趋于全局最优。

3. 【实验数据】

数据集：CityPosition1.mat、CityPosition2.mat、CityPosition3.mat。

数据集介绍：CityPosition1.mat 有 14*2 的数据组成，CityPosition2.mat 有 34*2 的数据组成，CityPosition3.mat 有 31*2 的数据组成，其中序号表示城市的序号，第一列表示当前序号的城市的 x 坐标，第二列表示当前序号的城市的 y 坐标。

4. 【实验原理】

(1) 模拟退火算法数学原理

模拟退火算法包含两个部分即 Metropolis 算法和退火过程,，分别对应内循环和外循环。外循环就是退火过程，将固体达到较高的温度（初始温度 T（0）），然后按照降温系数 alpha 使温度按照一定的比例下降，当达到终止温度 Tf 时，冷却结束，即退火过程结束。

Metropolis 算法是内循环，即在每次温度下，迭代 L 次，寻找在该温度下能量的最小值（即最优解）。

$$p=\begin{cases}1，E(n+1)<E(n)\\ e^{\frac{-(E(n+1)-E(n))}{T}},E(n+1)\geq E(n)\end{cases}$$

如上公式，假设当前时刻搜索的解为x_t，对应的系统能量(目标函数)为E_t，对搜索点施加随机扰动，产生新解x_{t+1}，相应地，系统能量为E_{t+1}，那么系统对搜索点从x_t到x_{t+1}转变的接受概率就为上公式。

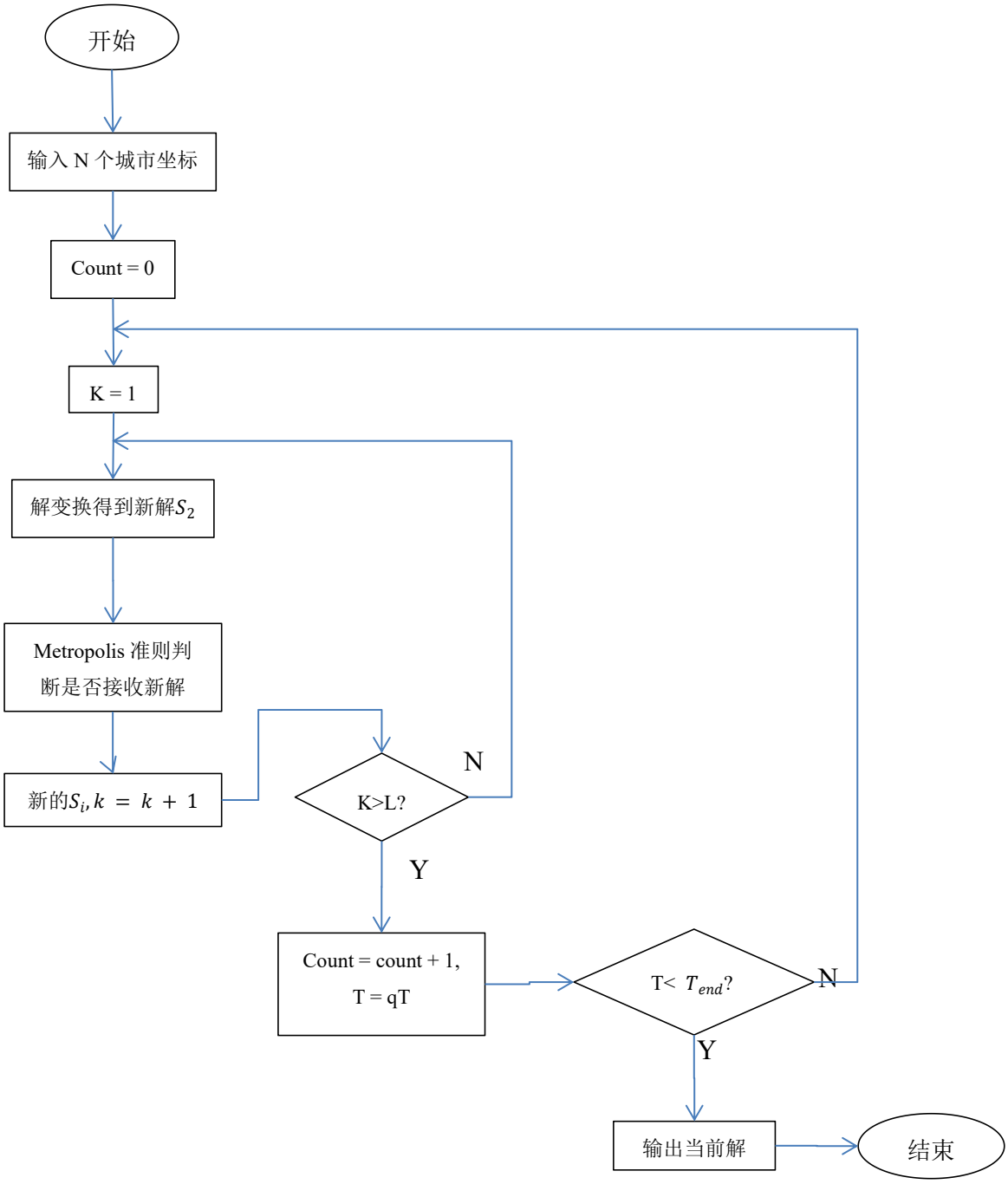
(2) 模拟退火算法不同温度下的情况

当温度很高时，落每个状态的概率是相等的；

当温度足够低时，等概率落入能量最低的状态；

当相同温度下，落入能量低的状态概率较大，落入能量高的状态概率较小；

当温度缓慢下降时，落入低能量状态的概率随之增加，落入高能量状态的概率随之减小。



(3) 使用可视化界面进行展示

使用 HMTL 对页面就进行布局，然后使用 JavaScript 实现模拟退火算法，解决 TSP 问题，可以进行输入指定城市数量和对应城市坐标，将 HTML 中输入的数据传给 JavaScript 中去，利用模拟退火算法解决 TSP 问题，最终将数据解决和最短路径返回回来展示在页面上，直观展示，最后可视化图表形式进行对比，鲜明看出参数对模拟退火算法的影响

5. 【实验过程】

在使用模拟退火算法解决 TSP 问题时，给出初始状态，也就是初始温度，终止温度，链长，降温速率

```
%%
tic
T0=1000;    % 初始温度

Tend=1e-3;  % 终止温度

L=500;      % 各温度下的迭代次数（链长）

q=0.95;     %降温速率
```

将输入导入，导入的数据是 CityPosition3 的数据;
对矩阵之间的距离进行计算

```
%%
D=Distanse(X);  %计算距离矩阵

N=size(D,1);     %城市的个数
```

对一开始随机产生一个初始路线，并将这个路径图画出来

```
%% 初始解

S1=randperm(N);  %随机产生一个初始路线

%% 画出随机解的路径图

DrawPath(S1,X)
pause(0.0001)
```

对于随机产生的路线的距离以及最后产生的总距离进行计算，并输出

```
%% 输出随机解的路径和总距离

disp('初始种群中的一个随机值:')

OutputPath(S1);
Rlength=PathLength(D,S1);

disp(['总距离: ',num2str(Rlength)]);
```

计算迭代次数

```
%% 计算迭代的次数 Time

syms x;
Time=ceil(double(solve(1000*(0.9)^x==Tend,x)));
```

```
count=0;          %迭代计数

Obj=zeros(Time,1);          %目标值矩阵初始化

track=zeros(Time,N);          %每代的最优路线矩阵初始化
```

根据模拟退火准则判断是否接受新解，对接收新的新解，我们需要记录下一路线的及其路程，反之，对于比上一最优解的那些值，我们不记录下一路线的及其路程

```
%% 迭代
while T0>Tend
    count=count+1;      %更新迭代次数
    temp=zeros(L,N+1);
    for k=1:L
        %% 产生新解
        S2=NewAnswer(S1);
        %% Metropolis 法则判断是否接受新解

        [S1,R]=Metropolis(S1,S2,D,T0);  %Metropolis 抽样算法

        temp(k,:)=[S1 R];           %记录下一路线的及其路程
    end
    %% 记录每次迭代过程的最优路线

    [d0,index]=min(temp(:,end)); %找出当前温度下最优路线
    if count==1 || d0<Obj(count-1)
        Obj(count)=d0;              %如果当前温度下最优路程小于上一路程则
记录当前路程
    else
        Obj(count)=Obj(count-1);%如果当前温度下最优路程大于上一路程则
记录上一路程
    end
    track(count,:)=temp(index,1:end-1);  %记录当前温度的最优路线

    T0=q*T0;      %降温
```

```
    fprintf(1,'%d\n',count)   %输出当前迭代次数
end
```

对现有的迭代图进行简单的优化

```
%% 优化过程迭代图
figure
plot(1:count,Obj)
xlabel('迭代次数')
ylabel('距离')
title('优化过程')
```

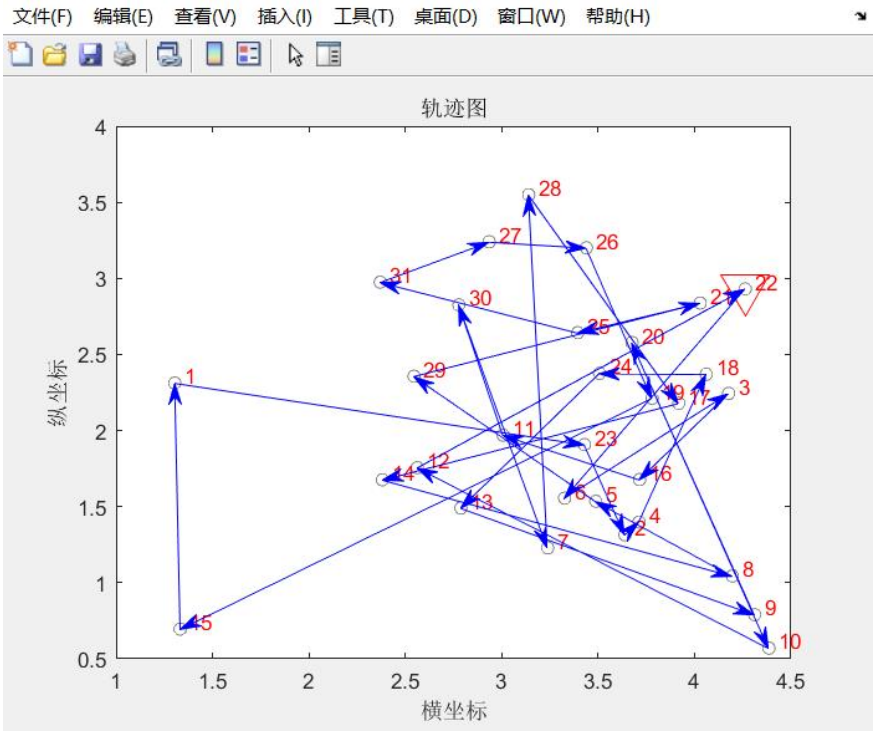
画出最优解的路径图并输出输出最优解的路线和总距离

```
%% 最优解的路径图
DrawPath(track(end,:),X)

%% 输出最优解的路线和总距离
disp('最优解:')
S=track(end,:);
p=OutputPath(S);
disp(['总距离: ',num2str(PathLength(D,S))]);
disp('------------------------------------------------------
------')
toc
```

6. 【实验结果及分析】

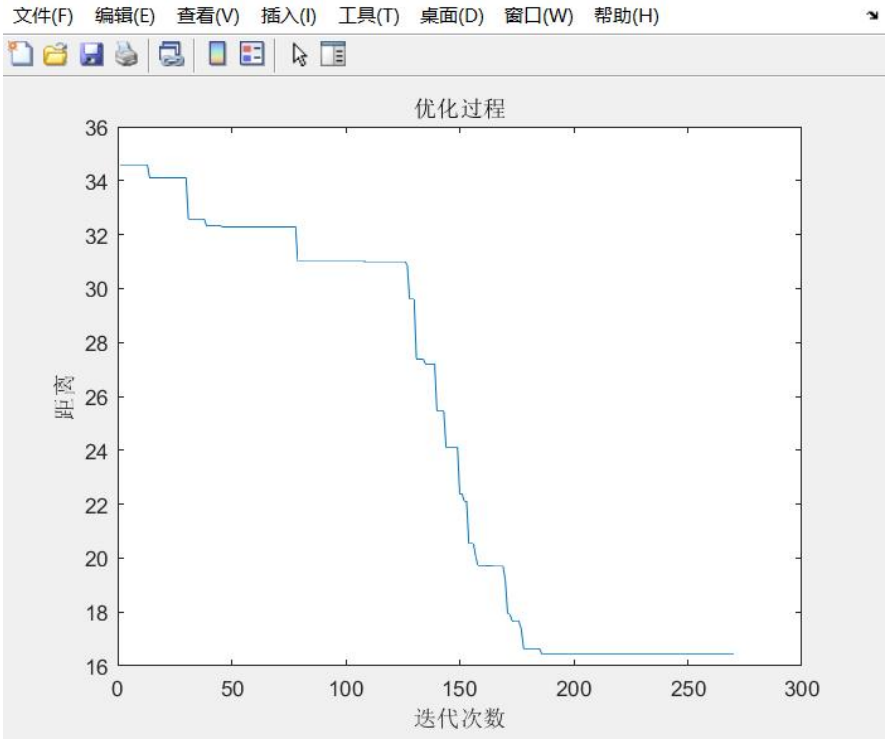
当生成一个随机解的时候，效果如下图所示：



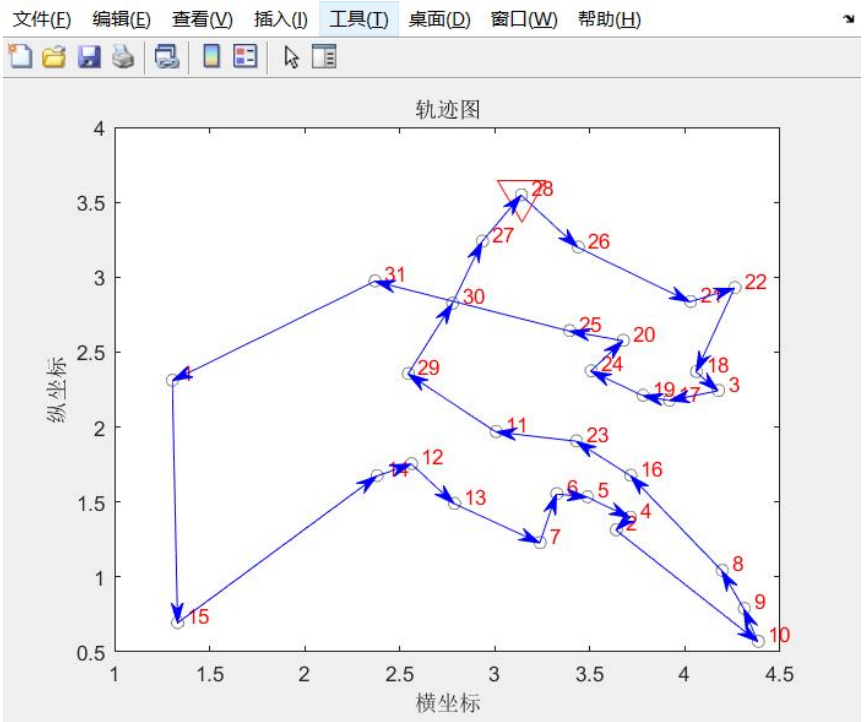
产生的路径对应的是

初始种群中的一个随机值:
25—>7—>2—>8—>29—>11—>4—>27—>9—>23—>17—>6—>28—>12—>14—>18—>10—>19—>3—>5—>22—>20—>26—>24—>31—>16—>15—>21—>30—>13—>1—>25
总距离：43.9429

最优解的迭代次数达到 270 次



最优解的路径图

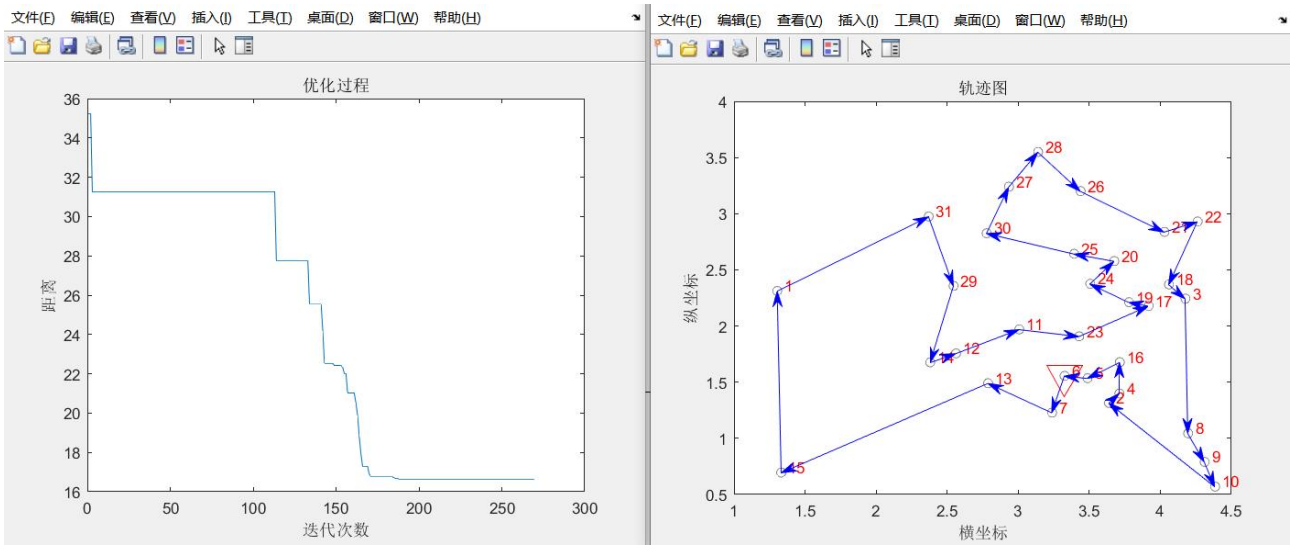


当生成一个最优解的时候，路径和总距离对应的是：

最优解:
28—>26—>21—>22—>18—>3—>17—>19—>24—>20—>25—>31—>1—>15—>14—>12—>13—>7—>6—>5—>4—>2—>10—>9—>8—>16—>23—>11—>29—>30—>27—>28
总距离：15.8758

（1）控制变量时，只改变链长为 800 时，保证其他变量不变
在当前条件下，生成一个随机解对应的路径和距离是：

初始种群中的一个随机值:
13—>27—>25—>4—>21—>12—>19—>6—>23—>29—>14—>2—>26—>17—>7—>11—>31—>16—>22—>18—>20—>24—>15—>1—>28—>8—>10—>3—>5—>30—>9—>13
总距离：41.4336



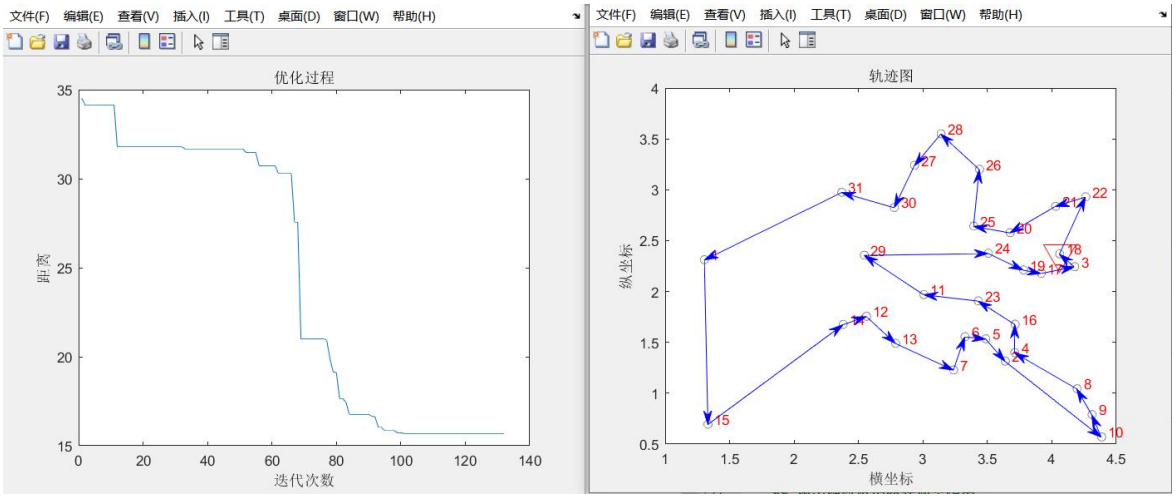
在当前条件下，生成一个最优解对应的路径和距离是：

最优解:
6—>7—>13—>15—>1—>31—>29—>14—>12—>11—>23—>17—>19—>24—>20—>25—>30—>27—>28—>26—>21—>22—>18—>3—>8—>9—>10—>2—>4—>16—>5—>6
总距离：16.6416

（2）控制变量时，只改变降温速率为 0.9 时，保证其他变量不变

在当前条件下，生成一个随机解对应的路径和距离是：

初始种群中的一个随机值:
20—>5—>17—>1—>21—>6—>13—>10—>25—>19—>4—>31—>29—>8—>2—>23—>15—>26—>16—>30—>18—>11—>14—>27—>28—>3—>12—>22—>7—>9—>24—>20
总距离：45.3869



在当前条件下，生成一个最优解对应的路径和距离是：

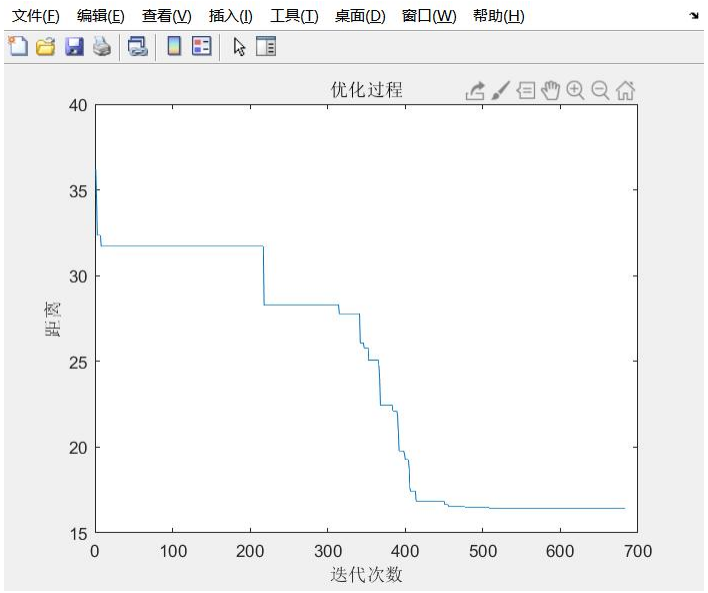
最优解:
18—>22—>21—>20—>25—>26—>28—>27—>30—>31—>1—>15—>14—>12—>13—>7—>6—>5—>2—>10—>9—>8—>4—>16—>23—>11—>29—>24—>19—>17—>3—>18
总距离：15.6912

（3）控制变量时，只改变降温速率为 0.98 时，保证其他变量不变

在当前条件下，生成一个随机解对应的路径和距离是：

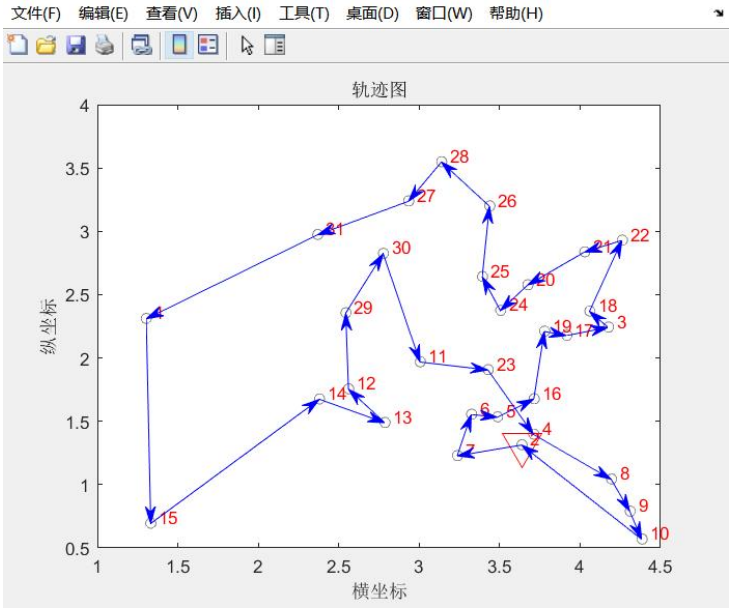
初始种群中的一个随机值:
27—>29—>11—>13—>22—>15—>31—>18—>1—>26—>23—>2—>25—>30—>14—>16—>12—>20—>17—>10—>28—>6—>3—>24—>21—>7—>19—>9—>4—>5—>8—>27
总距离：45.0136

当降温速率是 0.98 的时候，迭代次数高达 684 次



在当前条件下，生成一个最优解对应的路径和距离是：

最优解:
2—>7—>6—>5—>16—>19—>17—>3—>18—>22—>21—>20—>24—>25—>26—>28—>27—>31—>1—>15—>14—>13—>12—>29—>30—>11—>23—>4—>8—>9—>10—>2
总距离：16.4441



7. 【可视化展示】

使用 HTML 制做页面，在页面上可以自主输入选定城市数量、迭代次数、温度、降温速率，然后依次输入城市的坐标，第一栏是 X 轴坐标，第二栏是 Y 坐标；

可视化代码实现输入数据部分展示：

```
<div class="main">

    <div>
        请输入城市数量
        <input type="text" v-model.number="number">
```

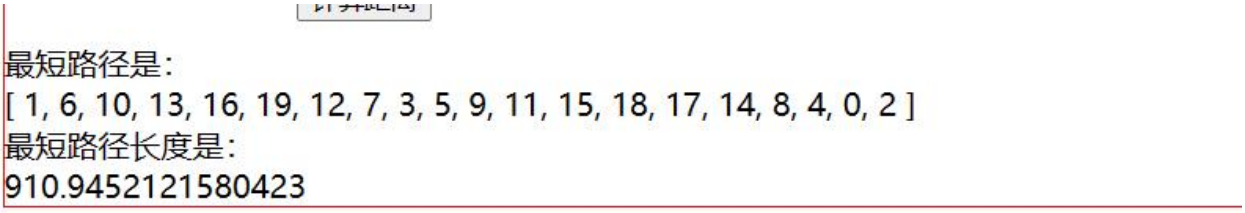
```
            请输入迭代次数
            <input type="text" v-model.number="iterations">
            请输入温度
            <input type="text" v-model.number="temperature">
            请输入降温速率
            <input type="text" v-model.number="coolingRate">
        </div>
        <br>
        <br>
        <tr>
            <td>城市 x 坐标</td>
            <td>城市 y 坐标</td>
        </tr>
        <br>
        <br>
        <div v-for="(x,y) in number">
            <tr>
                <td>
                    <input type="text" v-model="pathX[x]">
                </td>
                <td>
                    <input type="text" v-model="pathY[y]">
                </td>
            </tr>
        </div>
        <br>
        <button @click="calPath">计算距离</button>
    </div>

    <footer>
        最短路径是：
        <div v-show="this.result.path != []">{{this.result.path}}</div>
        最短路径长度是：
        <div v-show="this.result.path != []">{{this.result.length}}</div>
    </footer>
```

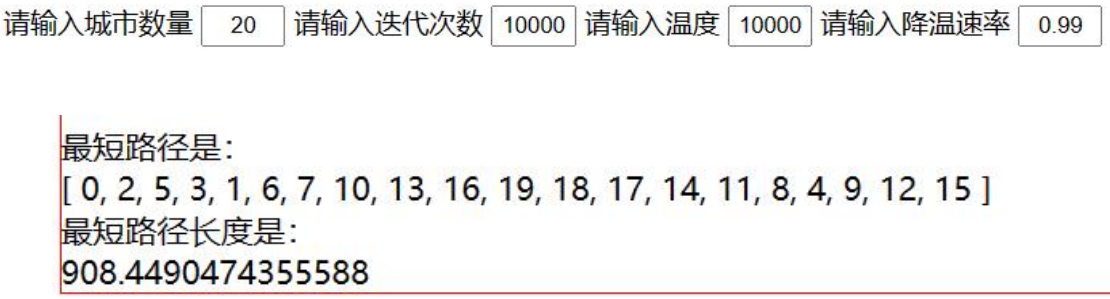
可视化效果展示：

输入一组数据，输出最有路径序列和最优路径的长度:

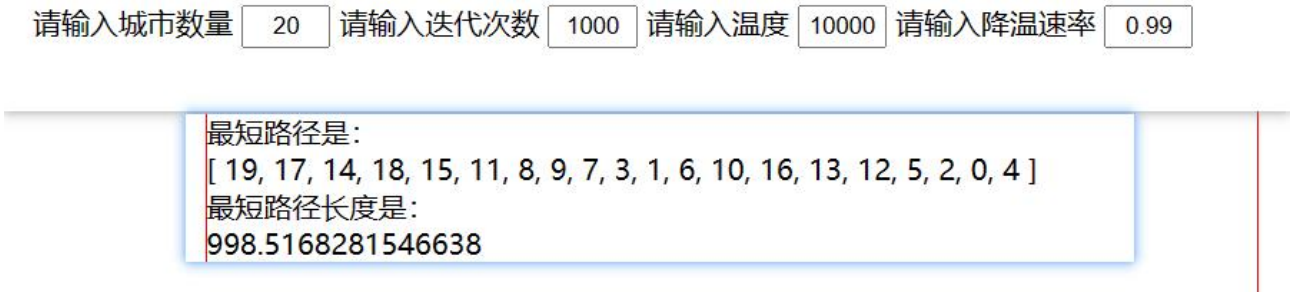
通过模拟退火算法之后得到的最优解：



通过修改迭代次数，重新计算这个最优解的情况
当迭代此时为 10000 的时候，其他参数不变：



当迭代次数为 1000 的时候其他变量保持不变的时候：



当迭代次数为 100 的时候其他变量保持不变的时候：

通过多组数据进行试验，不难看出来，当迭代次数少于一定的次数，这个最优路径长度比实际最优路径长度相差比较大

8. 【实验心得】

在这次实验中，将模拟退火算法从理论上应用于实际问题中，通过 TSP 这个案例，合理的将模拟退火算法从理论上过渡到实际中使用，进一步加深了对模拟退火算法的理解和使用，用图的形式表示这个整体过程，更加形象。通过这个迭代次数、随机生成的距离和最优解距离这些指标反馈出当前给定参数的最优性，当其他参数不变的时候随着降温速率在一定区间内上升，伴随着迭代次数也在增加；当其他参数不变的时候随着增加一定的链长，伴随着最优解的距离在一定程度上进行波动；通过这些参数的调整可以看出来，冷却速率较慢，则搜索时间较长，可以获得更优解，但是会花费大量时间，在合适的情况下使用模拟退火算法可以将问题很好的解决，最后通过可视化界面，输入城市数据和各个参数，清晰展示迭代次数对最优解的影响，这就是本次实验中我的收获。

9. 【附录】

代码

```
clc

clear

% 读取两个城市的坐标

load('CityPosition1.mat')

pos1=X;

load('CityPosition2.mat')
```

```
pos2 = X;
load('CityPosition3.mat')
pos3 = X;


 % 开始对城市位置进行可视化
x = pos1(:,1);
y = pos1(:,2);
plot(x,y,'o');
xlabel('城市横坐标');
ylabel('城市纵坐标');
grid on;


NIND = 100;                %种群大小
MAXGEN = 100;              %最大迭代次数
Pc = 0.9;                  %交叉概率，相当于基因遗传的时候染色体交叉
Pm = 0.05;                 %染色体变异
GGAP = 0.9;                %这个是代沟，通过遗传方式得到的子代数为父代数*GGAP
D = Distance(pos1);        %通过这个函数可以计算 i,j 两点之间的距离
N = size(D,1);             %计算有多少个坐标点
%%初始化种群
Chrom = InitPop(NIND,N);       %Chrome 代表的种群
%%画出随机解得路线图
DrawPath(Chrom(1,:),pos1)
pause(0.0001)
%输出随机解的路线和总距离
disp('初始种群中的一个随机值')
OutputPath(Chrom(1,:));%其中一个个体
```

```
Rlength = PathLength(D,Chrom(1,:));
disp(['总距离;',num2str(Rlength)]);
disp('~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~')
%优化
gen = 0;
trace = zeros(1,MAXGEN);
title('优化过程')
xlabel('迭代次数')
ylabel('当前最优解')
ObjV = PathLength(D,Chrom);%计算当前路线长度，即上面随机产生的那些个体路径
preObjV = min(ObjV);%当前最优解
while gen<MAXGEN
    %%计算适应度
    ObjV = PathLength(D,Chrom);        %计算路线长度
    pause(0.0001);
    preObjV = min(ObjV);
    trace(1,gen+1)=preObjV;
    %trace=[trace preObjV];
    FitnV = Fitness(ObjV);
    %选择
    SelCh = Select(Chrom,FitnV,GGAP);
    %交叉操作
    SelCh = Recombin(SelCh,Pc);
    %变异
    SelCh = Mutate(SelCh,D,Pm);
    Chrom = Reins(Chrom,SelCh,ObjV);
    gen = gen + 1;
```

```
end
%画出最优解的路线图
ObjV = PathLength(D,Chrom);          %计算路线长度
[minObjV,minInd] = min(ObjV);
DrawPath(Chrom(minInd(1),:),pos1)
figure();
plot([1:MAXGEN],trace(1,:));
%%输出最优解的路线和距离
disp('最优解:')
p = OutputPath(Chrom(minInd(1),:));
disp(['旅行商走过的总距离：',num2str(ObjV(minInd(1)))]);
disp('~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~')
```

可视化代码：

```
<!DOCTYPE html>
<html lang="en">
    <head>
        <meta charset="UTF-8">
        <title>模拟退火算法 TSP 问题</title>
    </head>

    <style>
        * {
            box-sizing: border-box;
        }

        #container {
            width: 1200px;
            /*height: 600px;*/
            margin: 20px auto;
            border: 1px solid red;
```

```
        }

        .main {
            width: 900px;
            margin: 10px auto;
        }

        input {
            width: 50px;
            height: 24px;
            text-align: center;
        }
    </style>
    <body>
        <div id="APP">
            <div id="container">
                <div>
                    <h1 style="text-align: center; color: coral">欢迎使用由 20 级计科 2 班彭清元设计的模拟退火算法 TSP 问题可视化展示</h1>
                </div>

                <div class="main">

                    <div>
                        请输入城市数量
                        <input type="text" v-model.number="number">
                        请输入迭代次数
                        <input type="text" v-model.number="iterations">
                        请输入温度
                        <input type="text" v-model.number="temperature">
                        请输入降温速率
                        <input type="text" v-model.number="coolingRate">

                    </div>
                    <br>
                    <br>
```

```
                <tr>
                    <td>城市 x 坐标</td>
                    <td>城市 y 坐标</td>
                </tr>
                <br>
                <br>
                <div v-for="(x,y) in number">
                    <tr>
                        <td>
                            <input type="text" v-model="pathX[x]">
                        </td>
                        <td>
                            <input type="text" v-model="pathY[y]">
                        </td>
                    </tr>
                </div>
                <br>
                <button @click="calPath">计算距离</button>
            </div>

            <footer>
                最短路径是：
                <div v-show="this.result.path != []">{{this.result.path}}</div>
                最短路径长度是：
                <div v-show="this.result.path != []">{{this.result.length}}</div>
            </footer>
        </div>
    </div>
    <script src="../../vue.js"></script>
    <script src="TSP.js"></script>
    <script>
        const vm = new Vue({
            el: "#APP",
            data: {
                number: 0,
                iterations: 0,
```

```
                temperature: 0,
                coolingRate: 0,
                pathX: [],
                pathY: [],
                path: [],
                result: [],
            },
            methods: {
                calPath() {

                    let len = this.pathX.length;

                    this.path = []

                    for (let i = 0; i < len - 1; i++) {
                        let arr = []
                        arr.push(this.pathX[i + 1])
                        arr.push(this.pathY[i])
                        this.path.push(arr)
                    }

                    this.result = main3(this.path, this.iterations, this.temperature, this.coolingRate)

                }
            }
        })
    </script>
</body>
</html>

// 定义城市坐标数组
var cities = [
    // [60, 200],
    // [180, 200],
    // [80, 180],
    // [140, 180],
    // [20, 160],
```

```
    // [100, 160],
    // [200, 160],
    // [140, 140],
    // [40, 120],
    // [100, 120],
    // [180, 100],
    // [60, 80],
    // [120, 80],
    // [180, 60],
    // [20, 40],
    // [100, 40],
    // [200, 40],
    // [20, 20],
    // [60, 20],
    // [160, 20]
];

// 计算两个城市之间的距离（欧氏距离）
function getDistance(city1, city2) {
    var xDistance = Math.abs(city1[0] - city2[0]);
    var yDistance = Math.abs(city1[1] - city2[1]);
    return Math.sqrt(xDistance * xDistance + yDistance * yDistance);
}

// 计算当前路径的总长度
function getPathLength(path) {
    var length = 0;
    for (var i = 0; i < path.length - 1; i++) {
        length += getDistance(cities[path[i]], cities[path[i + 1]]);
    }
    return length;
}

// 随机生成一个有效路径
function generatePath(numCities) {
    var path = [];
    for (var i = 0; i < numCities; i++) {
        path.push(i);
    }
    for (var i = 0; i < numCities - 1; i++) {
        var j = Math.floor(Math.random() * (numCities - i)) + i;
        var temp = path[i];
        path[i] = path[j];
        path[j] = temp;
    }
    return path;
```

```
}

// 模拟退火算法
function simulatedAnnealing(numCities, iterations, temperature, coolingRate) {
    var currentPath = generatePath(numCities);
    var currentLength = getPathLength(currentPath);
    var bestPath = currentPath.slice();
    var bestLength = currentLength;

    for (var i = 0; i < iterations; i++) {
        var newPath = currentPath.slice();

        // 随机交换两个城市的位置
        var j = Math.floor(Math.random() * numCities);
        var k = Math.floor(Math.random() * numCities);
        var temp = newPath[j];
        newPath[j] = newPath[k];
        newPath[k] = temp;

        var newLength = getPathLength(newPath);
        var delta = newLength - currentLength;

        if (delta < 0 || Math.exp(-delta / temperature) > Math.random()) {
            currentPath = newPath;
            currentLength = newLength;
            if (currentLength < bestLength) {
                bestPath = currentPath.slice();
                bestLength = currentLength;
            }
        }

        temperature *= coolingRate;
    }

    return {path: bestPath, length: bestLength};
}

// // 调用模拟退火算法求解 TSP 问题
// var result = simulatedAnnealing(cities.length, 100000, 10000, 0.99);
//
// console.log(result.path); // 输出最优路径
// console.log(result.length); // 输出最优路径的长度

function main3(arr,iterations,temperature,coolingRate) {
    // console.log(cities)
    // console.log(arr)
```

```
    cities = []
    for (let i = 0; i < arr.length; i++) {

        let array = [];
        array.push(parseInt(arr[i][0]))
        array.push(parseInt(arr[i][1]))
        cities.push(array)
    }

    //  调用模拟退火算法求解 TSP 问题
    // var result = simulatedAnnealing(cities.length, 100000, 10000, 0.99);
    var result = simulatedAnnealing(cities.length, iterations, temperature, coolingRate);

    // console.log(result.path); //  输出最优路径
    // console.log(result.length); //  输出最优路径的长度

    return result;
}
```

实验的过程不是消极的观察，而是积极的、有计划的探测，一个成功的实验需要的是眼光、勇气和毅力。

——曲阜师范大学名誉校长

诺贝尔奖获得者丁肇中