

**软件学院学生实验报告**

**实验课程名称：** 智能算法设计 **教师： 肖如良**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **实验项目名称** | **禁忌搜索算法解决旅行商的问题** | | | **实验成绩** |  |
| **学生姓名** | **高强** | **学 号** | **123032014010** | **年级专业班级** | **软三** |
| **小组成员** |  | | | **实验日期** | **2016.11.16** |

1. **实验目的**

ＴＳ算法通过模拟人类智能的记忆 机制，制造出一个“记忆装置”，即禁忌表。采用不同的禁忌策略已达到限制搜索过程陷入局部最优来避免迂回搜索。本实验基于禁忌搜索算法采用改进的新算法求解ＴＳＰ问题。

1. **实验软硬件环境**

Windows,eclipse,Jdk

1. **实验内容及要求**

TSP问题（Travelling Salesman Problem）描述的是：有一个旅行商人要拜访n个城市，他必须选择所要走的路径，路径的限制是每个城市只能拜访一次，而且最后要回到原来出发的城市。路径的选择目标是要求得的路径路程为所有路径之中的最小值。要求基于禁忌算法的基础上，通过改进算法来解决TSP的问题。

要求记录必要的过程和代码

1. **实验记录**

**一：实验步骤**

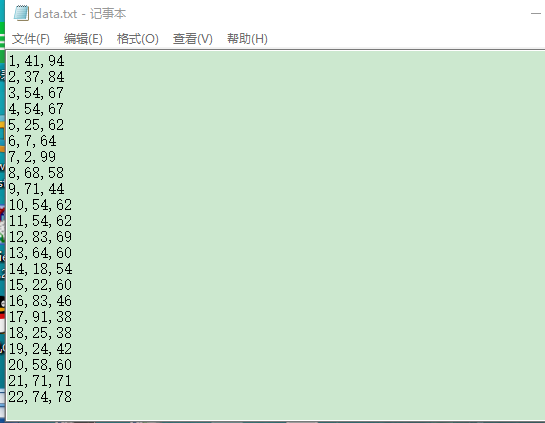
１）给定算法参数（候选解解个数、禁忌步数、终 止准则的循环步数等），随机产生初始解ｔｓｐｘ（数 组里包括路径和目标值），置禁忌表为空。

1. 判断算法终止条件是否满足？若是，则结束 算法并输出优化结果；否则，继续以下步骤。
2. 利用当前解的邻域函数产生其所有（或若 干）邻域解，并将其放入新数组tspslect（数组里包 括路径和目标值）。将新数组tspslect与随机产 生的初始解ｔspx进行一一比对，选出候选解。
3. 全新定义的藐视准则：按照路径的不相似度 和目标函数的相似度进行比例累加，百分比分数高 的为候选解。 对候选解判断藐视准则是否满足？若成立，则 用满足藐视准则的最佳状态tspnew替代tspx成 为新的当前解，即tspx＝tspnew，并用与tspnew 对应的禁忌对象替换最早进入禁忌表的禁忌对象， 同时用ｔｓｐｎｅｗ替换“best do far ”状态，然后转步骤

５）判断候选解对应的各对象的禁忌属性，选择 候选解集中非禁忌对象对应的最佳状态为新的当 前解，同时用与之对应的禁忌对象替换最早进入禁

1. ；否则，继续以下步骤。

二：数据录入



三：**代码部分**

import java.io.BufferedReader;

import java.io.FileInputStream;

import java.io.IOException;

import java.io.InputStreamReader;

import java.util.Random;

public class Tabu {

private int MAX\_GEN;// 迭代次数

private int N;// 每次搜索邻居个数

private int ll;// 禁忌长度

private int cityNum; // 城市数量，编码长度

private int[][] distance; // 距离矩阵

private int bestT;// 最佳出现代数

private int[] Ghh;// 初始路径编码

private int[] bestGh;// 最好的路径编码

private int bestEvaluation;

private int[] LocalGhh;// 当代最好编码

private int localEvaluation;

private int[] tempGhh;// 存放临时编码

private int tempEvaluation;

private int[][] jinji;// 禁忌表

private int t;// 当前代数

private Random random;

public Tabu() {

}

public Tabu(int n, int g, int c, int m) {

cityNum = n;

MAX\_GEN = g;

N = c;

ll = m;

}

private void init(String filename) throws IOException {

// 读取数据

int[] x;

int[] y;

String strbuff;

BufferedReader data = new BufferedReader(new InputStreamReader(

new FileInputStream(filename)));

distance = new int[cityNum][cityNum];

x = new int[cityNum];

y = new int[cityNum];

for (int i = 0; i < cityNum; i++) {

// 读取一行数据，数据格式1 6734 1453

strbuff = data.readLine();

// 字符分割

String[] strcol = strbuff.split(" ");

x[i] = Integer.valueOf(strcol[1]);// x坐标

y[i] = Integer.valueOf(strcol[2]);// y坐标

}

// 计算距离矩阵

// ，针对具体问题，距离计算方法也不一样，此处用的是att48作为案例，它有48个城市，距离计算方法为伪欧氏距离，最优值为10628

for (int i = 0; i < cityNum - 1; i++) {

distance[i][i] = 0; // 对角线为0

for (int j = i + 1; j < cityNum; j++) {

double rij = Math

.sqrt(((x[i] - x[j]) \* (x[i] - x[j]) + (y[i] - y[j])

\* (y[i] - y[j])) / 10.0);

// 四舍五入，取整

int tij = (int) Math.round(rij);

if (tij < rij) {

distance[i][j] = tij + 1;

distance[j][i] = distance[i][j];

} else {

distance[i][j] = tij;

distance[j][i] = distance[i][j];

}

}

}

distance[cityNum - 1][cityNum - 1] = 0;

Ghh = new int[cityNum];

bestGh = new int[cityNum];

bestEvaluation = Integer.MAX\_VALUE;

LocalGhh = new int[cityNum];

localEvaluation = Integer.MAX\_VALUE;

tempGhh = new int[cityNum];

tempEvaluation = Integer.MAX\_VALUE;

jinji = new int[ll][cityNum];

bestT = 0;

t = 0;

random = new Random(System.currentTimeMillis());

/\*

\* for(int i=0;i<cityNum;i++) { for(int j=0;j<cityNum;j++) {

\* System.out.print(distance[i][j]+","); } System.out.println(); }

\*/

}

// 初始化编码Ghh

void initGroup() {

int i, j;

Ghh[0] = random.nextInt(65535) % cityNum;

for (i = 1; i < cityNum;)// 编码长度

{

Ghh[i] = random.nextInt(65535) % cityNum;

for (j = 0; j < i; j++) {

if (Ghh[i] == Ghh[j]) {

break;

}

}

if (j == i) {

i++;

}

}

}

// 复制编码体，复制编码Gha到Ghb

public void copyGh(int[] Gha, int[] Ghb) {

for (int i = 0; i < cityNum; i++) {

Ghb[i] = Gha[i];

}

}

public int evaluate(int[] chr) {

// 0123

int len = 0;

// 编码，起始城市,城市1,城市2...城市n

for (int i = 1; i < cityNum; i++) {

len += distance[chr[i - 1]][chr[i]];

}

// 城市n,起始城市

len += distance[chr[cityNum - 1]][chr[0]];

return len;

}

// 邻域交换，得到邻居

public void Linju(int[] Gh, int[] tempGh) {

int i, temp;

int ran1, ran2;

for (i = 0; i < cityNum; i++) {

tempGh[i] = Gh[i];

}

ran1 = random.nextInt(65535) % cityNum;

ran2 = random.nextInt(65535) % cityNum;

while (ran1 == ran2) {

ran2 = random.nextInt(65535) % cityNum;

}

temp = tempGh[ran1];

tempGh[ran1] = tempGh[ran2];

tempGh[ran2] = temp;

}

// 判断编码是否在禁忌表中

public int panduan(int[] tempGh) {

int i, j;

int flag = 0;

for (i = 0; i < ll; i++) {

flag = 0;

for (j = 0; j < cityNum; j++) {

if (tempGh[j] != jinji[i][j]) {

flag = 1;// 不相同

break;

}

}

if (flag == 0)// 相同，返回存在相同

{

// return 1;

break;

}

}

if (i == ll)// 不等

{

return 0;// 不存在

} else {

return 1;// 存在

}

}

// 解禁忌与加入禁忌

public void jiejinji(int[] tempGh) {

int i, j, k;

// 删除禁忌表第一个编码，后面编码往前挪动

for (i = 0; i < ll - 1; i++) {

for (j = 0; j < cityNum; j++) {

jinji[i][j] = jinji[i + 1][j];

}

}

// 新的编码加入禁忌表

for (k = 0; k < cityNum; k++) {

jinji[ll - 1][k] = tempGh[k];

}

}

public void solve() {

int nn;

// 初始化编码Ghh

initGroup();

copyGh(Ghh, bestGh);// 复制当前编码Ghh到最好编码bestGh

bestEvaluation = evaluate(Ghh);

while (t < MAX\_GEN) {

nn = 0;

localEvaluation = Integer.MAX\_VALUE;

while (nn < N) {

Linju(Ghh, tempGhh);// 得到当前编码Ghh的邻域编码tempGhh

if (panduan(tempGhh) == 0)// 判断编码是否在禁忌表中

{

// 不在

tempEvaluation = evaluate(tempGhh);

if (tempEvaluation < localEvaluation) {

copyGh(tempGhh, LocalGhh);

localEvaluation = tempEvaluation;

}

nn++;

}

}

if (localEvaluation < bestEvaluation) {

bestT = t;

copyGh(LocalGhh, bestGh);

bestEvaluation = localEvaluation;

}

copyGh(LocalGhh, Ghh);

// 解禁忌表，LocalGhh加入禁忌表

jiejinji(LocalGhh);

t++;

}

System.out.println("最佳长度出现代数：");

System.out.println(bestT);

System.out.println("最佳长度");

System.out.println(bestEvaluation);

System.out.println("最佳路径：");

for (int i = 0; i < cityNum; i++) {

System.out.print(bestGh[i] + ",");

}

}

public static void main(String[] args) throws IOException {

System.out.println("Start....");

Tabu tabu = new Tabu(48, 1000, 200, 20);

tabu.init("c://data.txt");

tabu.solve();

}

}

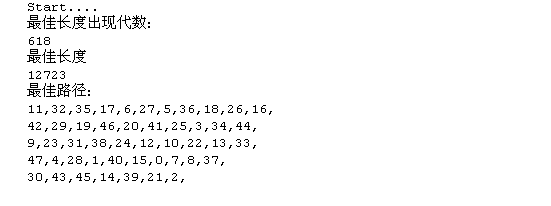
        tabu.init("c://data.txt");

        tabu.solve();

    }

}

**四：运算结果截图**



1. **实验小结（心得体会、遇到的问题及解决方式、未解决/需进一步研讨的问题或建议新实验方法等）**

**总结:**禁忌算法其主要特点是在搜索开始阶段，解的质量提高很快，随着搜索过程的继续，解的质量的提高速度逐渐放缓，甚至在很长的搜索阶段内解的质量没有太大提高，适合中小规模的NP问题求解，整体效率比较均衡