

计算机学院

人工智能

2020416046

2020级

周晓萌

人工智能实训

**曲阜师范大学实验报告**

**学院 计算机学院 年级、专业 20人工智能 学号 2020416046姓名 周晓萌**

**课程名称 人工智能实训 上课时间 第 1 节 — 第 10 节**

**实验日期 2022 年 9 月 3 日 星期 六 教师签名 雷玉霞 成绩**

**实验一 天使与野人过河问题**

**1．实验目的**

天使与野人过河问题：有三个天使和三个野人要过河，河里有一只船同时能容纳最多两个人。  
 假设天使和野人都会划船，但如果岸边（不包括船上）野人的数量比天使多，野人就要吃天使。  
问：怎么把三个野人和三个天使安全运输到对岸

1. **实验设备**
2. **Windows10 系统电脑**
3. **Dev C++**
4. **实验内容**

首先根据题目可以先设置一些规则，方便之后的解题。

规则一：两岸的天使都需要**大于或等于**野人的数量，**但是**，当天使的数量为0时，野人的数量可以大于天使。（**危险状态判定**）

规则二：去的时候一次**必须**转移两个人（因为转移一个人是没有意义的），回来的时候，可以一个人，也可以两个人。（**转移方案**）

规则三：同一种状态**不可以**转移两次。

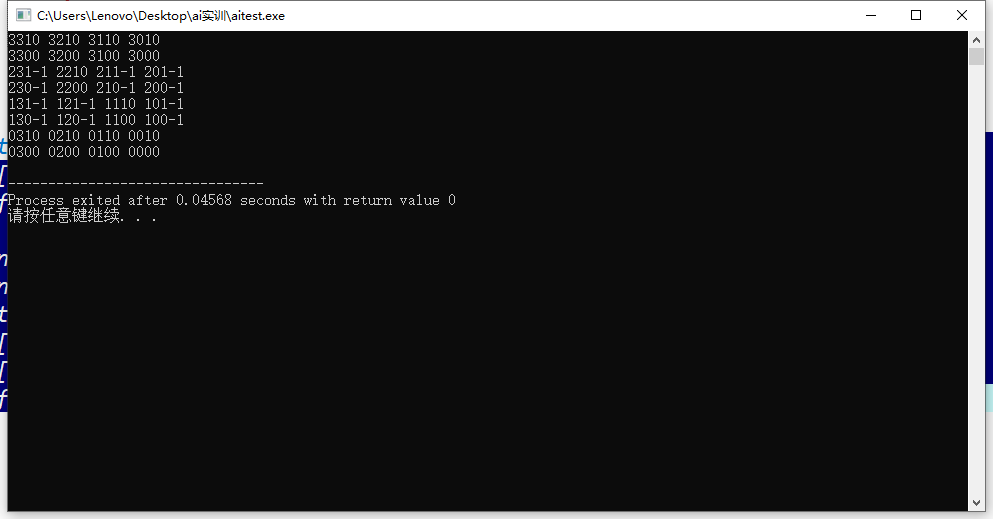
第二步，当设置了上述规则之后，可以对本题进行相应的分析。

是否可以只观察左岸的数据来判断是否处于安全状态？答案是可以的，因为当左岸状态固定时，右岸的状态也是可以确定的。

这时，可以设置一下状态的内容，以左岸为例，先是天使的数量，然后是野人的数量，其次是船的状态，最后是是否走过该状态。例如（3,3,1,0），此状态表示左岸有三个天使，三个野人，一艘船，并且没有走过次状态。

当这些前提都设置完成后，我确定使用**深度优先遍历（DFS）**来解决该题目。

第三步，先构建所有可能状态的矩阵，即初始化，当构建的时候，可以将危险状态标注出来，我将危险状态标为-1，放到是否走过的位置。例如（2,3,1，-1）。当构建完所有的状态后，可得如下矩阵。



当列举完状态后，便可以开始遍历。

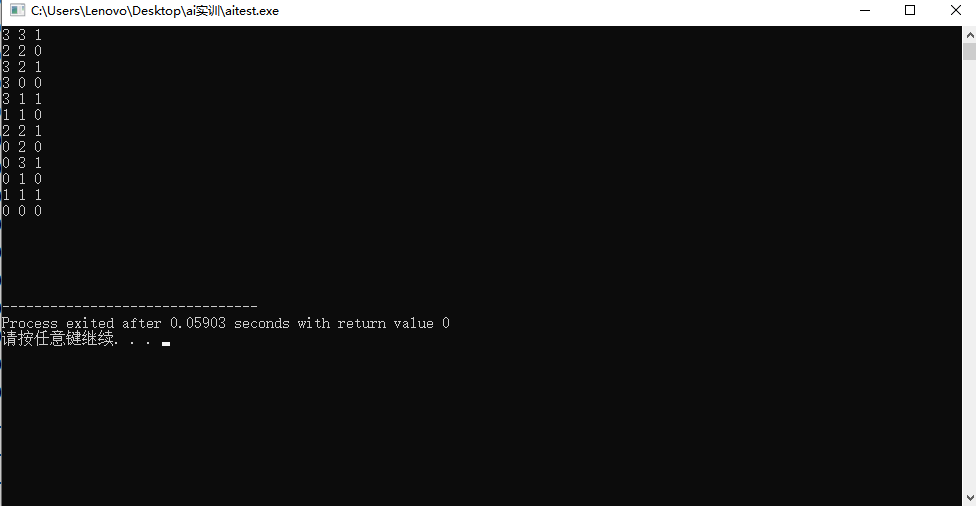
我将遍历分为了两类，一种是将船开到右岸，即为去。因为规则二，所以我只有三种方案，去两个天使，去两个野人和去一个天使和一个野人。当转移的时候需要判断是否可以转移，即转移不可使左岸人数少于0。

同理，另一种为将船开到左岸，即为回。因为规则二，这时有五种方案，为回两个天使，回两个野人，回一个天使和一个野人，回一个天使和回一个野人。

这样，这八种情况就是所有的转移方案。

这时就可以开始递归操作，当递归结束时，即可得到答案。

但是，此时会出现一个问题，深度优先搜索只可以得到一种答案，以为每走一步都需要标记当前位置，所以走完之后就会结束递归。如下所示。



这时，我想到了一种解决办法，可以先进行一次深度优先遍历，当遍历到（0,0,0,0）时，这时还会有的状态没有遍历到，这些没有遍历到的状态即为分支点的新分支，我们可以通过这些节点找到其父亲节点，然后通过枚举这些节点的新分支将所有结果输出。

最终结果如下：



1. **实验心得**

通过本次实验，我掌握到了深度优先搜索的具体应用，并且对分析问题有了较为深刻的理解，通过设置规则将问题具体化，方便后续的问题解答。

但同时，我的解决方案也有着许多问题，首先是深度优先搜索并不是很适合本题目，使用了两次深度优先搜索才将该题解决，存在着时间上的冗余，若使用宽度优先搜索可以大大的节省搜索的时间。其次是未能将图形化的界面展示出来，内容不够直观。

附录：（代码部分）

#include <iostream>

#include <cstdio>

#include <queue>

#include <cstring>

#include <algorithm>

#include <stack>

using namespace std;

#define N 100

typedef pair<int , int > PII;

#define x first

#define y second

int people\_n = 3; //angle and people number

int ship\_n = 2; // ship

int a[N][N][4]; // state

int stay[4];

int flag = 0; // zhao dao

PII q[100];//shu chu

int cnt = 0 ; //ji shu

void init();//chu shi

void dfs();//shen sou

int judge(int a1, int b1, int c1, int d1);//pan duan

int main()

{

// chu shi hu

init();

// for(int i = 0 ; i <= people\_n\*2 + 1 ; i ++)

// {

// for(int j = 0 ; j <= people\_n ; j ++ )

// {

// for(int k = 0 ; k < 4 ; k ++)

// {

// printf("%d",a[i][j][k]);

// }

// printf(" ");

// }

// printf("\n");

// }

stay[0] = people\_n , stay[1] = people\_n , stay[2] = 1 , stay[3] = 0;

q[0].x = 3 ,q[0].y = 3;

dfs();

init();

cnt = 0;

stay[0] = people\_n , stay[1] = people\_n , stay[2] = 1 , stay[3] = 0;

q[0].x = 3 ,q[0].y = 3;

a[3][1][3] = -1;

dfs();

}

void init()

{

int num\_a = people\_n;

int num\_b = people\_n;

int num\_c = 1;

int num\_d = 0;

int jishu = 0;

for(int i = 0 ; i <= people\_n\*2 + 1 ; i ++ )

{

jishu++;

if(jishu == 3)

{

num\_a -- ;

jishu = 1;

}

num\_b = people\_n;

for(int j = 0 ; j <= people\_n ; j ++ )

{

for(int k = 0 ; k < 4 ; k ++ )

{

if(k == 0) a[i][j][k] = num\_a;

if(k == 1) a[i][j][k] = num\_b;

if(k == 2 && i%2 == 0) a[i][j][k] = num\_c;

if(k == 2 && i%2 == 1) a[i][j][k] = !num\_c;

if(k == 3) a[i][j][k] = num\_d;

if((a[i][j][0] < a[i][j][1] && a[i][j][0] != 0 ||

a[i][j][0] > a[i][j][1] && a[i][j][0] != people\_n) && k == 3 )

a[i][j][k] = -1; // weixian

}

num\_b -- ;

}

}

}

void dfs()

{

if(stay[0] == 0 && stay[1] == 0 && stay[2] == 0 && stay[3] == 0)

{

flag ++ ;

// if(flag==1)

for(int i = 0 ; i <= cnt ; i ++ )

{

printf("%d %d %d\n", q[i].x,q[i].y,(i + 1) % 2);

}

printf("\n");

// for(int i = 0 ; i <= people\_n\*2 + 1 ; i ++)

// {

// for(int j = 0 ; j <= people\_n ; j ++ )

// {

// for(int k = 0 ; k < 4 ; k ++)

// {

// printf("%d",a[i][j][k]);

// }

// printf(" ");

// }

// printf("\n");

// }

cnt -- ;

stay[0] = q[cnt].x;

stay[1] = q[cnt].y;

stay[2] = !stay[2];

// if(a[(people\_n - stay[0]) \* 2 + !stay[2]][people\_n - stay[1]][3] == 1) //shi fang zou guo de

// a[(people\_n - stay[0]) \* 2 + !stay[2]][people\_n - stay[1]][3] = 0 ;

return ; // tiaochu

}

if(stay[2] == 1) // go

{

if(judge(stay[0],stay[1],stay[2],stay[3]))

{

q[cnt].x = stay[0];

q[cnt].y = stay[1];

if(stay[0] - 2 >= 0 && judge(stay[0] - 2,stay[1],!stay[2],stay[3]))

{

cnt ++ ;

a[(people\_n - stay[0]) \* 2 + !stay[2]][people\_n - stay[1]][3] = 1; // biaoji

// a[(people\_n - stay[0]) \* 2 + 1][people\_n - stay[1]][3] = 1; // biaoji

stay[0] = stay[0] - 2;

stay[2] = 0;

dfs();

}

}

else

{

cnt -- ;

stay[0] = q[cnt].x;

stay[1] = q[cnt].y;

stay[2] = !stay[2];

if(a[(people\_n - stay[0]) \* 2 + !stay[2]][people\_n - stay[1]][3] == 1) //shi fang zou guo de

a[(people\_n - stay[0]) \* 2 + !stay[2]][people\_n - stay[1]][3] = 0 ;

return ;

}

if(judge(stay[0],stay[1],stay[2],stay[3]))

{

q[cnt].x = stay[0];

q[cnt].y = stay[1];

if(stay[0] - 1 >= 0 && stay[1] - 1 >= 0 && judge(stay[0] - 1,stay[1] - 1,!stay[2],stay[3]))

{

cnt ++ ;

a[(people\_n - stay[0]) \* 2 + !stay[2]][people\_n - stay[1]][3] = 1; // biaoji

// a[(people\_n - stay[0]) \* 2 + 1][people\_n - stay[1]][3] = 1; // biaoji

stay[0] = stay[0] - 1;

stay[1] = stay[1] - 1;

stay[2] = 0;

dfs();

}

}

else

{

cnt -- ;

stay[0] = q[cnt].x;

stay[1] = q[cnt].y;

stay[2] = !stay[2];

if(a[(people\_n - stay[0]) \* 2 + !stay[2]][people\_n - stay[1]][3] == 1) //shi fang zou guo de

a[(people\_n - stay[0]) \* 2 + !stay[2]][people\_n - stay[1]][3] = 0 ;

return ;

}

// else return;

if(judge(stay[0],stay[1],stay[2],stay[3]))

{

q[cnt].x = stay[0];

q[cnt].y = stay[1];

if(stay[1] - 2 >= 0 && judge(stay[0],stay[1] - 2,!stay[2],stay[3]))

{

cnt ++ ;

a[(people\_n - stay[0]) \* 2 + !stay[2]][people\_n - stay[1]][3] = 1; // biaoji

// a[(people\_n - stay[0]) \* 2 + 1][people\_n - stay[1]][3] = 1; // biaoji

stay[1] = stay[1] - 2;

stay[2] = 0;

dfs();

}

}

else

{

cnt -- ;

stay[0] = q[cnt].x;

stay[1] = q[cnt].y;

stay[2] = !stay[2];

if(a[(people\_n - stay[0]) \* 2 + !stay[2]][people\_n - stay[1]][3] == 1) //shi fang zou guo de

a[(people\_n - stay[0]) \* 2 + !stay[2]][people\_n - stay[1]][3] = 0 ;

return ;

}

}

else // back

{

if(judge(stay[0],stay[1],stay[2],stay[3]))

{

q[cnt].x = stay[0];

q[cnt].y = stay[1];

if(stay[0] + 2 <= people\_n && judge(stay[0] + 2,stay[1],!stay[2],stay[3]))

{

cnt ++ ;

a[(people\_n - stay[0]) \* 2 + !stay[2]][people\_n - stay[1]][3] = 1; // biaoji

// a[(people\_n - stay[0]) \* 2 + 1][people\_n - stay[1]][3] = 1; // biaoji

stay[0] = stay[0] + 2;

stay[2] = 1;

dfs();

}

}

else

{

cnt -- ;

stay[0] = q[cnt].x;

stay[1] = q[cnt].y;

stay[2] = !stay[2];

if(a[(people\_n - stay[0]) \* 2 + !stay[2]][people\_n - stay[1]][3] == 1) //shi fang zou guo de

a[(people\_n - stay[0]) \* 2 + !stay[2]][people\_n - stay[1]][3] = 0 ;

return ;

}

if(judge(stay[0],stay[1],stay[2],stay[3]))

{

q[cnt].x = stay[0];

q[cnt].y = stay[1];

if(stay[0] + 1 <= people\_n && stay[1] + 1 <= people\_n && judge(stay[0] + 1,stay[1] + 1,!stay[2],stay[3]))

{

cnt ++ ;

a[(people\_n - stay[0]) \* 2 + !stay[2]][people\_n - stay[1]][3] = 1; // biaoji

// a[(people\_n - stay[0]) \* 2 + 1][people\_n - stay[1]][3] = 1; // biaoji

stay[0] = stay[0] + 1;

stay[1] = stay[1] + 1;

stay[2] = 1;

dfs();

}

}

else

{

cnt -- ;

stay[0] = q[cnt].x;

stay[1] = q[cnt].y;

stay[2] = !stay[2];

if(a[(people\_n - stay[0]) \* 2 + !stay[2]][people\_n - stay[1]][3] == 1) //shi fang zou guo de

a[(people\_n - stay[0]) \* 2 + !stay[2]][people\_n - stay[1]][3] = 0 ;

return ;

}

if(judge(stay[0],stay[1],stay[2],stay[3]))

{

q[cnt].x = stay[0];

q[cnt].y = stay[1];

if(stay[1] + 2 <= people\_n && judge(stay[0],stay[1] + 2,!stay[2],stay[3]))

{

cnt ++ ;

a[(people\_n - stay[0]) \* 2 + !stay[2]][people\_n - stay[1]][3] = 1; // biaoji

// a[(people\_n - stay[0]) \* 2 + 1][people\_n - stay[1]][3] = 1; // biaoji

stay[1] = stay[1] + 2;

stay[2] = 1;

dfs();

}

}

else

{

cnt -- ;

stay[0] = q[cnt].x;

stay[1] = q[cnt].y;

stay[2] = !stay[2];

if(a[(people\_n - stay[0]) \* 2 + !stay[2]][people\_n - stay[1]][3] == 1) //shi fang zou guo de

a[(people\_n - stay[0]) \* 2 + !stay[2]][people\_n - stay[1]][3] = 0 ;

return ;

}

// else return;

if(judge(stay[0],stay[1],stay[2],stay[3]))

{

q[cnt].x = stay[0];

q[cnt].y = stay[1];

if(stay[0] + 1 <= people\_n && judge(stay[0] + 1,stay[1],!stay[2],stay[3]))

{

cnt ++ ;

a[(people\_n - stay[0]) \* 2 + !stay[2]][people\_n - stay[1]][3] = 1; // biaoji

// a[(people\_n - stay[0]) \* 2 + 1][people\_n - stay[1]][3] = 1; // biaoji

stay[0] = stay[0] + 1;

stay[2] = 1;

dfs();

}

}

else

{

cnt -- ;

stay[0] = q[cnt].x;

stay[1] = q[cnt].y;

stay[2] = !stay[2];

if(a[(people\_n - stay[0]) \* 2 + !stay[2]][people\_n - stay[1]][3] == 1) //shi fang zou guo de

a[(people\_n - stay[0]) \* 2 + !stay[2]][people\_n - stay[1]][3] = 0 ;

return ;

}

if(judge(stay[0],stay[1],stay[2],stay[3]))

{

q[cnt].x = stay[0];

q[cnt].y = stay[1];

if(stay[1] + 1 <= people\_n && judge(stay[0],stay[1],!stay[2],stay[3]))

{

cnt ++ ;

a[(people\_n - stay[0]) \* 2 + !stay[2]][people\_n - stay[1]][3] = 1; // biaoji

// a[(people\_n - stay[0]) \* 2 + 1][people\_n - stay[1]][3] = 1; // biaoji

stay[1] = stay[1] + 1;

stay[2] = 1;

dfs();

}

}

else

{

cnt -- ;

stay[0] = q[cnt].x;

stay[1] = q[cnt].y;

stay[2] = !stay[2];

if(a[(people\_n - stay[0]) \* 2 + !stay[2]][people\_n - stay[1]][3] == 1) //shi fang zou guo de

a[(people\_n - stay[0]) \* 2 + !stay[2]][people\_n - stay[1]][3] = 0 ;

return ;

}

}

}

int judge(int a1, int b1, int c1, int d1)

{

int i = (people\_n - a1) \* 2 + !c1;

int j = people\_n - b1;

if(a[i][j][3] == -1 || a[i][j][3] == 1)

{

return 0; // no

}

return 1; // yes

}

**曲阜师范大学实验报告**

**学院 计算机学院 年级、专业 20人工智能 学号 2020416046姓名 周晓萌**

**课程名称 人工智能实训 上课时间 第 1 节 — 第 10 节**

**实验日期 2022 年 9 月 10 日 星期 六 教师签名 雷玉霞 成绩**

**实验二 知识提取与合并**

**1．实验目的**

1. 掌握分析文本字段的能力，从中提取重要的知识组成标准的文字。
2. 掌握在不同文本之间整合相似或相同的信息，进行合并处理。

**2．实验设备**

1. 一台装有windows10系统的笔记本电脑。
2. 集成Anaconda 11.0.8编译环境和python3.0环境。
3. **实验成员**

周晓萌 贾宝玺 范树玉 纪旭乐

**3. 实验内容**

1. **分析问题**
2. 在老师的实验任务讲解和演示中我们可以得知，本次实验的实验任务是处理略微工整的一段文字，在文字中提取出重要、有用的知识，将这些提取出来的知识用一段新的文字串联起来，形成一段标准的文字段。同时经过我的初步分析，需要处理的此段文字，一般都带有强烈的主谓宾关系，都是形如“...的...是...”的语句，所以在一般的处理中，可以根据关键词“的”来区分句子的主语和谓语，用关键词“是”来区分句子的谓语和宾语，以此来达到文字处理的过程。
3. 对于知识合并部分，实际是将不同的，已经经过知识提取处理的文本进行合并，寻找这些文本中的相同点或相似点，合并为一段文字。对于文本中的不同点，则分别用语句来呈现，不采取合并形式。
4. 样例使用的文本处理对象是学校信息类的方面，我们经过商讨后决定使用音乐类的文本信息，因为音乐类文本信息普遍具有标准性信息，例如作词人、作曲人、音乐所属专辑等等。所以在音乐类文本中提取这些信息也可以比一般的文字稍微提高一下难度。

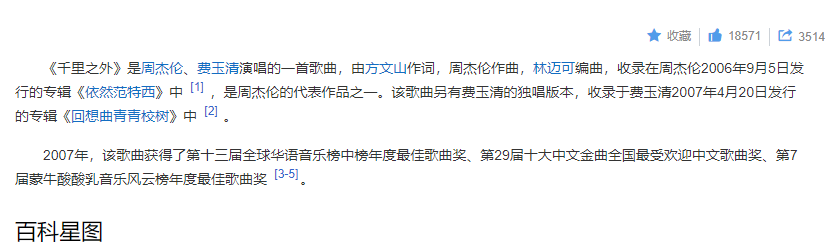


图1 音乐类文本处理信息展示

1. **本程序创新点**
2. 基于对本实验任务的分析和讨论，我们小组决定拔高自己编写的程序的水平，设计更为智能的程序，就知识提取这一点，我们决定不给予被处理文本更高的工整性和标准性，即我们所要处理的对象是较为随机的语句段，不全是带有强烈主谓宾关系，这样更考验程序实现的鲁棒性。面对这样的随机语句段，在程序运行处理成功后，生成一个标准的知识提取文本。
3. 本程序支持四个语句段的同时提取，基于音乐类文本信息，我们可以同时提取四个文本中的演唱作词作曲等信息，同时在处理完成后，进行知识合并，将四个处理信息中相同的作者摘出来列举形成文本。
4. 因为不是对带有强烈主谓宾关系的语句进行处理，所以我们需要用到自然语言中的中文分词，中文分词指的是将一个汉字序列切分成一个一个单独的词，分词就是将连续的字序列按照一定的规范重新组合成词序列的过程。利用一个中文词库，确定汉字之间的关联概率。
5. **程序分工及原理**

经小组讨论，我们决定安排纪旭乐同学做音乐类信息的查找和收集，范树玉同学做爬虫爬取数据集需要的音乐样本，周晓萌同学负责知识提取和知识合并部分的代码书写，贾宝玺同学负责交互式UI界面的设计和实验报告的书写。

* 音乐类信息的查找和收集：

为方便信息的统一和便于处理性，我们选择在百度百科上搜索歌曲名，从而得到整首歌的数据信息，从百度百科搜索到的信息每个歌曲的信息也是不同，所以保证了数据的差异性，提高了数据做知识提取的难度。

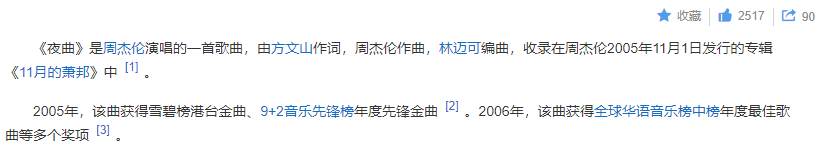


图2 音乐类信息查找和收集

* 爬虫爬取音乐样本数据集：

因为用到了中文分词，所以需要选择合适的中文词库，来提高汉字之间的关联概率，但是对于音乐方面的中文词库非常稀少，所以我们选择另一种途径，即除了分词外，添加自定义词组来当做音乐词库，所以需要搜集音乐的数据，因此用到python爬虫。用爬虫爬取各大音乐网站的词组，采集例如“《弯弯的月亮》”，“刘欢”这样的词组音乐信息，集合成一个txt文档作为音乐词库调用。

爬虫技术主要是根据网页的源码进行处理。要对网页源码有清晰地认识，要求编程人员具有HTML的基础，在对应的网页源码中进行网页语言的处理。

依据本实验的特殊性以及对自然语言处理的特殊性，在爬网页信息的过程中曾编写对于链家网页房源的具体信息。基于链家房源网页的网页代码的规整性，爬取的过程中，只要发现HTML的层级的标签以及基于对于网页源码的检查，就能很快的提取出关键词以及信息。

在实际小组内合作中，我们小组内成员仔细探讨了任务分工。最终发现，在网页的提取过程中，可针对歌曲类网页进行特殊型的爬取即可。因此，基于对于网页不同信息的浏览，我们发现歌曲的信息爬取以及信息的组合是可以满足任务的分配。

最终，确定了周杰伦等歌手的歌曲相关资料的爬取，为任务完成打下基础。

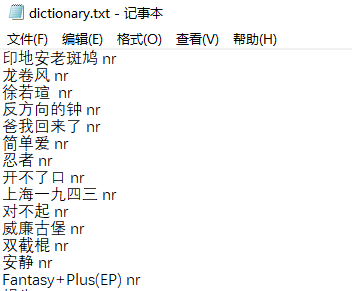


图3 爬虫爬取音乐样本数据集

* 知识提取和知识合并：

**知识提取**：

## 当我们获得一段文本时，想要将其中的关键信息提取出来的话，首先要进行分词处理，我们采用jieba库进行分词。

1. 第一步安装jieba：

jieba支持pip或者conda安装，直接在命令行执行：

pip install jieba

1. 第二步确定使用模式：

Jieba有全模式和精确模式等模式，这里我们采用精确模式（默认即为精确模式）。

1. 第三步导入自定义词库：

因为我们要对歌名和人们进行分词处理，极其容易出现分词错误，所以再这里要先提前创建自定义词库，我们的自定义词库是通过爬虫将网络上的歌名、歌手、作词人、作曲人和编曲人爬取下来，最终创建自定义词库，具体详细内容可见爬虫爬取音乐样本数据集。当我们将jieba的配置都设定好之后，开始分词，这时就可以看到分词的效果很好：

* 原始文本：《晴天》是周杰伦作词、作曲、编曲并演唱的歌曲，收录在周杰伦2003年7月31日发行的专辑《叶惠美》中。
* 分词之后：《 晴天 》 是 周杰伦 作词 、 作曲 、 编曲 并 演唱 的 歌曲 ，收录 在 周杰伦 2003 年 7 月 31 日 发行 的专辑 《 叶惠美 》 中 。

1. 然后，要建立关键词之间的联系。我们将“作词”、“作曲”、“编词”、“编曲”和“演唱”之类的词作为关键词，当一句话之中出现这些词的时候，我们就需要寻找这一句话中出现的人名，当同时找到这些信息时，即可建立关系。同时，我们还需要将歌曲名和专辑名找出来，只要是被书名号包围的即为歌曲名与专辑名。当找到这些信息后，我们即可把这些关键信息链接起来，输出出来。效果如下：

* 原始文本：《晴天》是周杰伦作词、作曲、编曲并演唱的歌曲，收录在周杰伦2003年7月31日发行的专辑《叶惠美》中。
* 处理之后：

《晴天》：

周杰伦演唱

周杰伦作词

周杰伦作曲

周杰伦编曲

收录于《叶惠美》

将这些操作重复进行即可实现多组文本的提取。

**知识合并**：

当我们获得到一些处理后的信息后，可以将这些信息中重叠的地方合并起来，例如，周杰伦演唱了《晴天》，周杰伦演唱了《青花瓷》，我们可以把这些信息和起来成：周杰伦演唱了：《晴天》，《青花瓷》。这样就可以将所有的信息合起来。

* 交互式UI界面：

软件的UI界面我们选择使用python第三方库WX，制作了两个界面用于显示信息，第一个界面用于显示整个软件的功能和框架，正上方有软件的名称“中文提取与整合”，在第一个界面的左侧添加了四个文本框用于填写需要处理的提取字段，并在下方设有一个知识提取的交互按钮，点击即可将已填入的文本提取字段进行知识提取，并在右侧对应的文本框里显示出来。

为防止用户无法区分需要提取字段框还是显示框，我们分别在两侧加了提示信息，并编写代码使得显示框无法输入文字，即将文本框设为只读模式，提高软件的可使用性。在第一个界面的右侧下方也设置了一个交互按钮，点击可以跳转第二个界面，第二个界面是用来显示出知识合并后的数据，点击交互按钮的同时会对四个文本框进行合并整合处理，最终将处理完成的数据通过文本框呈现出来。

1. **软件效果展示**

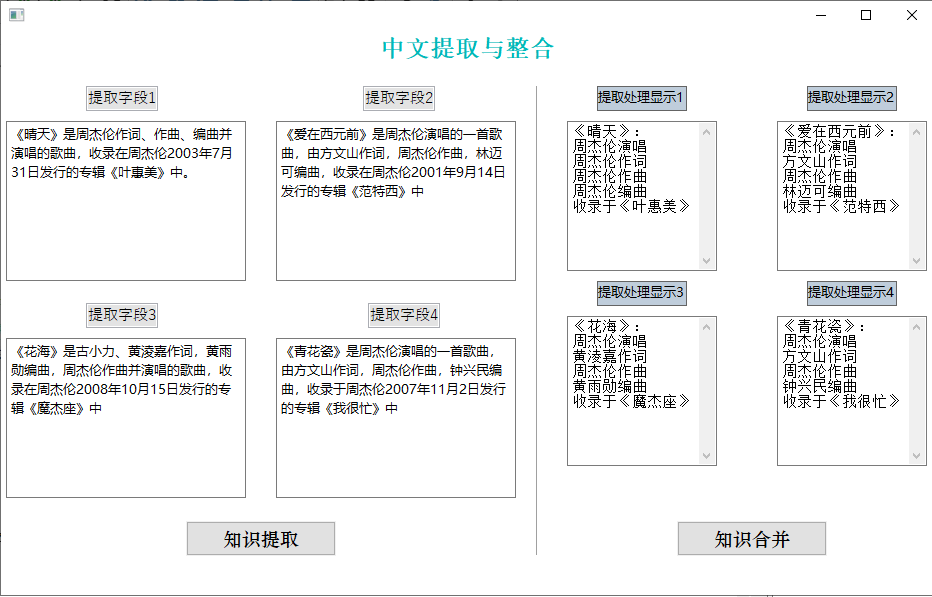


图4 知识提取功能展示

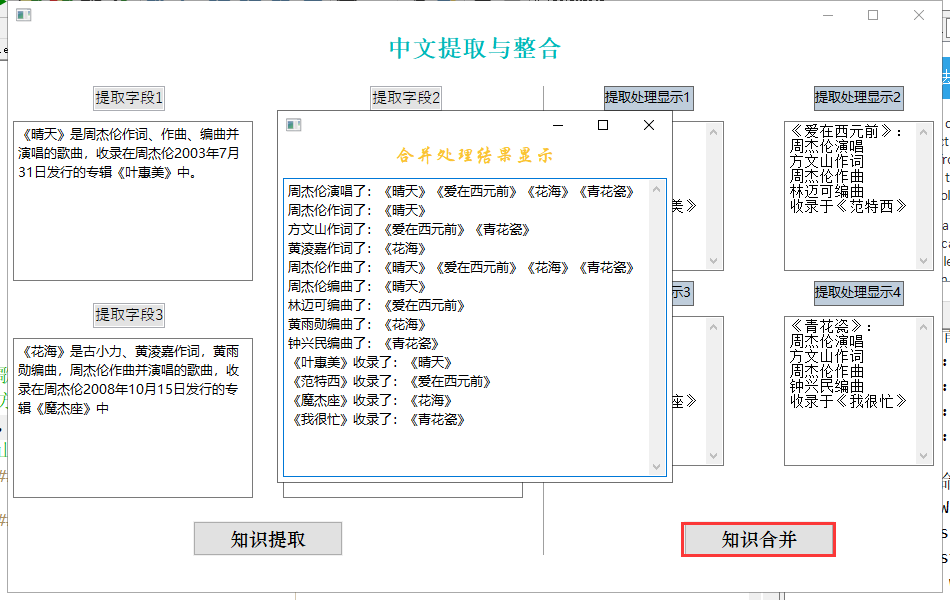


图5 知识合并功能展示

**4．实验心得**

本次实验考察自然语言处理中的知识提取和知识合并，在自然语言处理这一方面我们普遍接触还是比较少的，一般都是跟据某一个关键词或者标点符号进行划分，这次通过本实验重点学习了一下中文分词，初步认识了自然语言处理的知识和方法，联想到前几年出的QQ小冰和苹果siri等人工智能化的机器人交互，都是基于对自然语言的处理，不过融合了语音处理等高级算法，比我们目前学习到的要智能很多。另一点本次实验是以小组为单位做的实验，我们小组成员经常在一起交流问题，所以这次实验的分工配合也是极为默契熟练，每个人在自己熟悉的方面都得到了锻炼和提升，我们充分利用了个人擅长的领域进行分工协作，并将各人所学知识发挥组合使得软件成品在代码方面更为全面，实验成果更为成功。

附录：部分知识提取代码:

for jishu in range(4):

if jishu == 0 and wenben1 != "":

words = pseg.cut(wenben1)

flag\_wen = 1

if jishu == 1 and wenben2 != "":

words = pseg.cut(wenben2)

flag\_wen = 1

if jishu == 2 and wenben3 != "":

words = pseg.cut(wenben3)

flag\_wen = 1

if jishu == 3 and wenben4 != "":

words = pseg.cut(wenben4)

flag\_wen = 1

if flag\_wen == 1:

for word ,pos in words:

gequword.append(word)

gequpos.append(pos)

longword = longword + 1

for i in range(longword):

if gequpos[i] == "x":

if gequword[i] == "《":

while 1:

i = i + 1

if gequpos[i] == "nr" and name6 == "" and name5 != "" : #找到专辑名了

name6 = gequword[i]

if gequword[i] == "》": #结束

break

if gequpos[i] == "nr" and name5 == "" : #找到歌曲名了

name5 = gequword[i]

if gequword[i] == "》": #结束

break

if gequword[i] == "，":

j = i

cnt = 0

while 1:

j = j - 1

if j <= 1 :

cnt = cnt + 1

j = i

if gequword[j] == "，":

j = i

cnt = cnt + 1

if cnt >= 4:

cnt = 0

# flag1 = 0

# flag2 = 0

# flag3 = 0

# flag4 = 0

break

if gequword[j] == "演唱" and flag1 == 0:

flag1 = 1

j = i

while 1:

if j <= 0 :

cnt = cnt + 1

break

j = j - 1

if gequpos[j] == "nr":

name1 = gequword[j]

j = i

cnt = cnt + 1

break

if gequword[j] == "作词" and flag2 == 0:

flag2 = 1

j = i

while 1:

if j <= 0 :

cnt = cnt + 1

break

j = j - 1

if gequpos[j] == "nr":

name2 = gequword[j]

j = i

cnt = cnt + 1

break

if gequword[j] == "作曲" and flag3 == 0:

flag3 = 1

j = i

while 1:

if j <= 0 :

cnt = cnt + 1

break

j = j - 1

if gequpos[j] == "nr":

name3 = gequword[j]

j = i

cnt = cnt + 1

break

if gequword[j] == "编曲" and flag4 == 0:

flag4 = 1

j = i

while 1:

if j <= 0 :

cnt = cnt + 1

break

j = j - 1

if gequpos[j] == "nr":

name4 = gequword[j]

j = i

cnt = cnt + 1

break

if flag\_wen != 1:

if jishu == 0:

self.show1.SetValue("")

if jishu == 1:

self.show2.SetValue("")

if jishu == 2:

self.show3.SetValue("")

if jishu == 3:

self.show4.SetValue("")

if flag\_wen == 1:

if jishu == 0:

cunchu1.append(name1)

cunchu2.append(name2)

cunchu3.append(name3)

cunchu4.append(name4)

cunchu5.append(name5)

cunchu6.append(name6)

print(cunchu1[0],cunchu2[0],cunchu3[0],cunchu4[0],cunchu5[0],cunchu6[0])

self.tian1 = "《"+name5 + "》：\n" + name1+"演唱\n"+ name2+"作词\n"+name3+"作曲\n"+name4+"编曲\n"+"收录于" + "《"+name6 + "》"

self.show1.SetValue(self.tian1)

print("《"+name5 + "》：" , name1+"演唱",name2+"作词",name3+"作曲",name4+"编曲","收录于" + "《"+name6 + "》" )

if jishu == 1:

cunchu1.append(name1)

cunchu2.append(name2)

cunchu3.append(name3)

cunchu4.append(name4)

cunchu5.append(name5)

cunchu6.append(name6)

self.tian2 = "《"+name5 + "》：\n" + name1+"演唱\n"+ name2+"作词\n"+name3+"作曲\n"+name4+"编曲\n"+"收录于" + "《"+name6 + "》"

self.show2.SetValue(self.tian2)

print("《"+name5 + "》：" , name1+"演唱",name2+"作词",name3+"作曲",name4+"编曲","收录于" + "《"+name6 + "》" )

if jishu == 2:

cunchu1.append(name1)

cunchu2.append(name2)

cunchu3.append(name3)

cunchu4.append(name4)

cunchu5.append(name5)

cunchu6.append(name6)

self.tian3 = "《"+name5 + "》：\n" + name1+"演唱\n"+ name2+"作词\n"+name3+"作曲\n"+name4+"编曲\n"+"收录于" + "《"+name6 + "》"

self.show3.SetValue(self.tian3)

print("《"+name5 + "》：" , name1+"演唱",name2+"作词",name3+"作曲",name4+"编曲","收录于" + "《"+name6 + "》" )

if jishu == 3:

cunchu1.append(name1)

cunchu2.append(name2)

cunchu3.append(name3)

cunchu4.append(name4)

cunchu5.append(name5)

cunchu6.append(name6)

self.tian4 = "《"+name5 + "》：\n" + name1+"演唱\n"+ name2+"作词\n"+name3+"作曲\n"+name4+"编曲\n"+"收录于" + "《"+name6 + "》"

self.show4.SetValue(self.tian4)

print("《"+name5 + "》：" , name1+"演唱",name2+"作词",name3+"作曲",name4+"编曲","收录于" + "《"+name6 + "》" )

#print("《"+name5 + "》：" , name1+"演唱",name2+"作词",name3+"作曲",name4+"编曲","收录于" + "《"+name6 + "》" )

**曲阜师范大学实验报告**

**学院 计算机学院 年级、专业 20人工智能 学号 2020416046姓名 周晓萌**

**课程名称 人工智能实训 上课时间 第 1 节 — 第 10 节**

**实验日期 2022 年 9 月 17 日 星期 六 教师签名 雷玉霞 成绩**

**实验三 基于遗传算法的旅行商问题**

**1．实验目的**

1. 通过遗传算法，解决旅行商问题。

**2．实验设备**

1. 一台装有windows10系统的笔记本电脑。
2. matlab环境。
3. **实验成员**

周晓萌

**3. 实验内容**

1. **问题背景**
2. 旅行商问题，即TSP问题(Travelling Salesman Problem)又译为旅行推销员问题、货郎担问题，是数学领域中著名问题之一。假设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。
3. 本次实验的实验任务是通过遗传算法，解决旅行商问题。遗传算法（Genetic Algorithm ,简称GA），是模拟达尔文的遗传选择和自然淘汰的生物进化过程的计算机算法，它由美国Holland教授1957年提出。
4. 遗传算法作为一种新的全局优化搜索算法，以其简单通用、鲁棒性强、适合并行处理及应用范围广等显著特点，奠定了它作为21世纪关键智能计算之一的地位。
5. **处理过程**

遗传算法计算优化的过程就如同生物学上生物遗传进化的过程，主要有三个基本操作（或算子）：

1. 选择（Selection）
2. 交叉（Crossover）
3. 变异（Mutation）

第一步，进行编码操作，我们采用二进制形式来解决编码问题，即将某个变量值代表的个体表示为一个{0, 1}二进制串。将城市序号以二进制形式进行编码。

第二步，定义适应函数和适应值，这里的适应函数就是求所有城市的距离之和的函数，而适应值则是距离之和。

第三步，确定选择标准，这里用适应值反比来作为入选概率。

第四步，产生种群，将入选概率大的个体选入种群，淘汰概率小的个体，并用概率最大的个体补入种群，得到与原群体大小同样的种群。

第五步，交叉，交叉也就是将一组染色体上对应基因段的交换得到新的染色体，然后得到新的染色体组，组成新的群体。在这里就是随机交换了两条路径的顺序，产生新的顺序。

第六步，变异，变异就是通过一个小概率改变染色体位串上的某个基因。在这里就是随机改变路径顺序上的一个点。

第七步，终止条件，遗传算法的终止条件有两类常见条件：

（1）采用设定最大（遗传）代数的方法，一般可设定为 50代，此时就可能得出最优解．此种方法简单易行，但可能不是很精确

（2） 根据个体的差异来判断，通过计算种群中基因多样性测度，即所有基因位相似程度来进行控制

最后，通过上述操作，得到了如下结果：

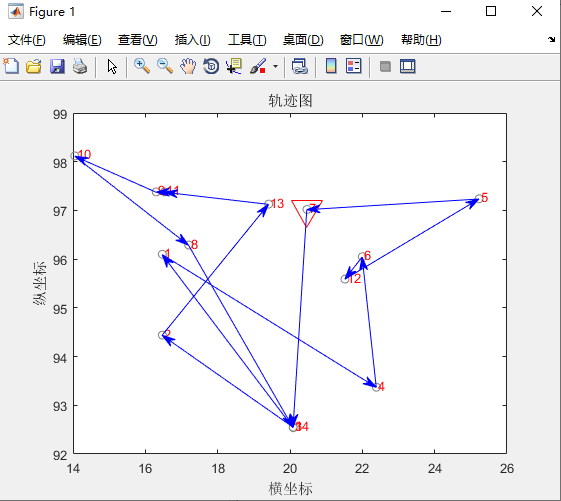


图1 随机解路径图

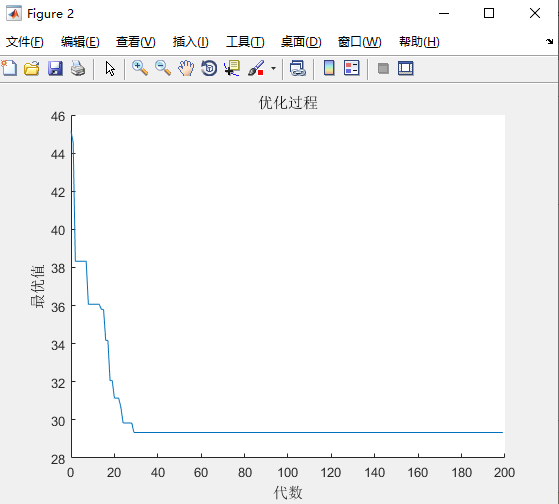


图2 优化迭代结果

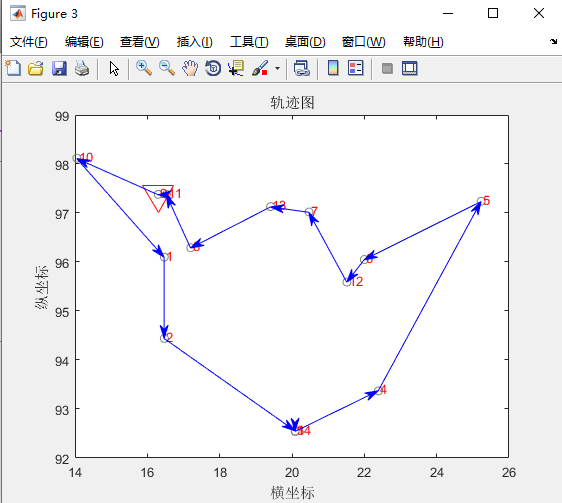


图3 最优解路径图

**4．实验心得**

本次实验考察了如何将人工智能中的遗传算法应用于实际问题，通过合适的策略将旅行商问题解决。其实，解决旅行商问题的途径有很多，例如贪心算法，动态规划等等，相较于这些算法，遗传算法具有以下优点：

1：能够求出优化问题的全局最优解。

2：优化结果与初始条件无关。

3：算法独立于求解域。

4：具有较强的鲁棒性。

5：适合于求解复杂的优化问题。

6：应用较为广泛。

但同时，遗传算法也具有一些缺点：

1：收敛速度慢。

2：局部搜索能力差。

3：控制变量较多。

4：无确定的终止准则。

所以，在合适的情况下使用遗传算法可以将问题很好的解决，这就是本次实验中我的收获。

**曲阜师范大学实验报告**

**学院 计算机学院 年级、专业 20人工智能 学号 2020416046姓名 周晓萌**

**课程名称 人工智能实训 上课时间 第 1 节 — 第 10 节**

**实验日期 2022 年 9 月 24 日 星期 六 教师签名 雷玉霞 成绩**

**实验四 基于模拟退火算法的旅行商问题**

**1．实验目的**

1. 通过模拟退火算法，解决旅行商问题。

**2．实验设备**

1. 一台装有windows10系统的笔记本电脑。
2. matlab环境。
3. **实验成员**

周晓萌

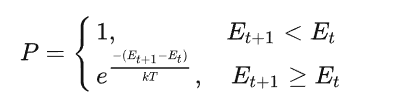
**3. 实验内容**

1. **问题背景**
2. 旅行商问题，即TSP问题(Travelling Salesman Problem)又译为旅行推销员问题、货郎担问题，是数学领域中著名问题之一。假设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。
3. 本次实验的实验任务是通过模拟退火算法，解决旅行商问题。模拟退火算法(Simulated Annealing,SA)最早的思想是由N. Metropolis等人于1953年提出。1983年,S. Kirkpatrick等成功地将退火思想引入到组合优化领域。它是基于Monte-Carlo 迭代求解策略的一种随机寻优算法，其出发点是基于物理中固体物质的退火过程与一般组合优化问题之间的相似性。
4. **处理过程**

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,结合一定的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。这里的“一定的概率”的计算参考了金属冶炼的退火过程，这也是模拟退火算法名称的由来。将温度T当作控制参数，目标函数值f视为内能E，而固体在某温度T时的一个状态对应一个解xi，然后算法试图随着控制参数T的降低，使目标函数f(内能E)也逐渐降低，直至趋于全局最小值（退火中低温时的最低能量状态），就像金属退火过程一样。

1. **模拟退火数学原理**

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，如果新解比当前解更优，则接受新解，否则基于Metropolis准则判断是否接受新解。接受概率为：



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

1. **模拟退火流程**

算法实质分两层循环，在任一温度水平下，随机扰动产生新解，并计算目标函数值的变化，决定是否被接受。由于算法初始温度比较高，这样，使E增大的新解在初始时也可能被接受，因而能跳出局部极小值，然后通过缓慢地降低温度，算法就最终可能收敛到全局最优解，具体流程为：

1. 令T=T0,表示开始退火的初始温度，随机产生一个初始解x0,并计算对应的目标函数值E(x0);
2. 令T=kT,其中k取值0到1之间，为温度下降速率；
3. 对当前解xt施加随机扰动，在其邻域内产生一个新解xt+1，并计算对应的目标函数值E(xt+1),计算

ΔE=E(xt+1)−E(xt)

1. 若ΔE<0,接受新解作为当前解，否则按照概率e−ΔE/kT判断是否接受新解；
2. 在温度T下，重复L次扰动和接受过程，即执行步骤3和4；
3. 判断温度是否达到终止温度水平，若是则终止算法，否则返回步骤2.
4. **最终效果**

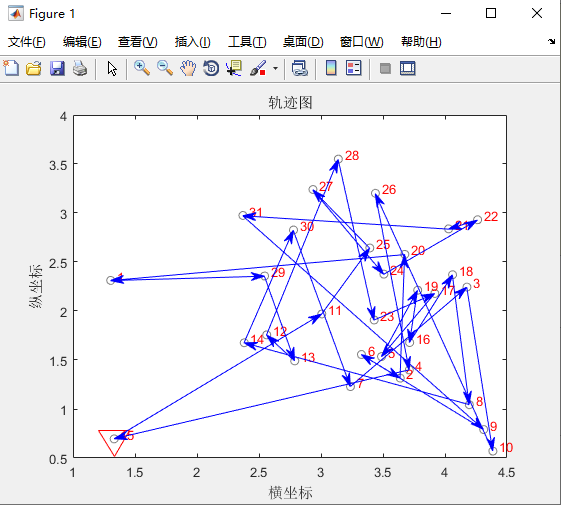


图1 随机解路径图

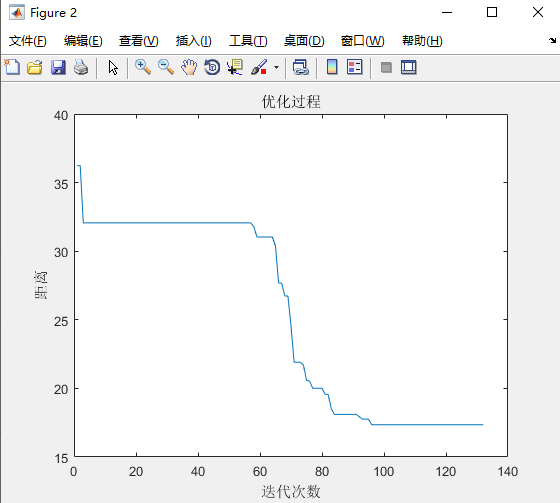


图2 优化迭代结果

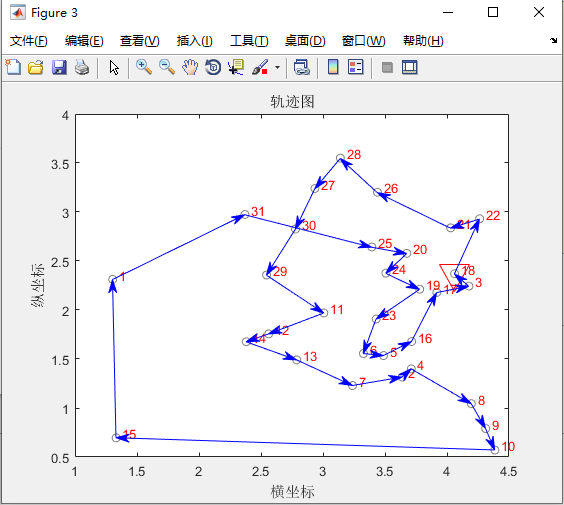


图3 最优解路径图

**4．实验心得**

本次实验考察了如何将人工智能中的模拟退火算法应用于实际问题，通过合适的策略将旅行商问题解决。其实，解决旅行商问题的途径有很多，例如贪心算法，动态规划等等，相较于这些算法，遗传算法具有以下优点：

1：可以突破爬山算法的局限性，获得全局最优解（以一定的概率接受较差解，从而跳出局部最优解）。

2：初始解与最终解都是随机选取的，它们毫无关联，因此具有很好的鲁棒性，即抵御外界不稳定因素的能力。

但同时，遗传算法也具有一些缺点：

1：其最优解常常受迭代次数k的影响，若k值越大，则搜索时间越长，获得的最优解更可靠；k值越小，则搜索时间越短，有可能就跳过了最优解。

2：模拟退火算法受温度冷却速率的影响，若冷却速率较慢，则搜索时间较长，可以获得更优解，但是会花费大量时间；如果冷却速率过快，可以较快的搜索到更优的解，但也有可能直接跳过最优解。

所以，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，这就是本次实验中我的收获。

