太原理工大学
TAIYUAN UNIVERSITY OF TECHNOLOGY

本科实验报告

课程名称：《人工智能原理》

实验项目：启发式搜索、产生式系统、归结原理、遗传算法

实验地点：实验楼 110

专业班级：物联网 1501 学号：2015001965

学生姓名：高磊

指导教师：强彦

2018 年 4 月 26 日

实验一 启发式搜索

一 实验目的

熟悉和掌握启发式搜索的定义、估价函数和算法过程，并利用 A 算法求解九宫问题，理解求解流程和搜索顺序。

二 实验内容以及原理

实验内容：用启发式搜索方法求解迷宫问题。

启发式搜索就是在状态空间中的搜索对每一个搜索的位置进行评估，得到最好的位置，再从这个位置进行搜索直到目标。这样可以省略大量无谓的搜索路径，提高了效率。在启发式搜索中，对位置的估价是十分重要的。采用了不同的估价可以有不同的效果。我们先看看估价是如何表示的。

启发中的估价是用估价函数表示的，如：最佳优先搜索的最广为人知的形式称为A*搜索(发音为“A星搜索”).它把到达节点的耗散g(n)和从该节点到目标节点的消耗h(n)结合起来对节点进行评价：f(n)=g(n)+h(n)因为以g(n)给出了从起始节点到节点n的路径耗散，而h(n)是从节点n到目标节点的最低耗散路径的估计耗散值，因此f(n)=经过节点n的最低耗散解的估计耗散.这样，如果我们想要找到最低耗散解，首先尝试找到g(n)+h(n)值最小的节点是合理的。可以发现这个策略不只是合理的:倘若启发函数h(n)满足一定的条件，A*搜索既是完备的也是最优的。

如果把A*搜索用于Tree-Search,它的最优性是能够直接分折的。在这种情况下，如果h(n)是一个可采纳启发式--也就是说，倘若h(n)从不会过高估计到达目标的耗散--A*算法是最优的。可采纳启发式天生是最优的，因为他们认为求解问题的耗散是低于实际耗散的。因为g(n)是到达节点n的确切耗散，我们得到一个直接的结论：f(n)永远不会高估经过节点n的解的实际耗散。

启发算法有： 蚁群算法，遗传算法、模拟退火算法等，蚁群算法是一种来自大自然的随机搜索寻优方法，是生物界的群体启发式行为，现已陆续应用到组合优化、人工智能、通讯等多个领域。蚁群算法的正反馈性和协同性使其可用于分布式系统，隐含的并行性更使之具有极强的发展潜力。从数值仿真结果来看，它比目前风行一时的遗传算法、模拟退火算法等有更好的适应性。

三 实验仪器

Intel(R) Core(TM) i5-4210M CPU @ 2.6GHz 2.6GHz
8G RAM
Windows10 专业版 1803 Linux 子系统 Ubuntu 16.04
VS code +；Cmder；g++5.4

四 实验步骤/设计实现

1. 确定状态的表示与状态空间
2. 设计估计函数，使用估测点与终点的欧几里得距离作为表示，估计函数值越小，状

态越优。

3. 确定拓展规则：每次选择临近状态的最优解。

五 实验代码

```
//main.cpp
// Created by gaolex on 18-4-19.
#include <iostream>
#include "heuristicSearch.h"
using namespace std;
int main() {
    myMap m;
    m.init("map1.map");
    m.findPath();
    for(auto i:m.m){
        for(auto j:i)
            cout<<j;
        cout<<endl;
    }
    return 0;
}
```

```
//map1.map
10
10
**********
*s*      * *
* *      * *
* **     * *
*   *       *
*   *       *
*   *       *
*   *   e   *
*           *
**********
```

```
//heuristicSearch.h
// Created by gaolex on 18-4-19.
#ifndef AI_HOMEWORK_HEURISTICSEARCH_H
#define AI_HOMEWORK_HEURISTICSEARCH_H
#include <string>
#include <vector>
class position{
public:
    position(){};
    bool operator==(position p){return p.x==x&&p.y==y;}
    position(int i,int j):x{i},y{j}{}
    int x,y;
};
class myMap{
public:
    int evalutionFunction(int ,int);
    void init(std::string);
    int length,width;
    position posCurrent,posStart,posEnd;
    std::vector<std::string> m;
    int findPath();
};
```

```
#endif //AI_HOMEWORK_HEURISTICSEARCH_H
```

```
//heuristicSearch.c
// Created by gaolex on 18-4-19.
#include "heuristicSearch.h"
#include <cmath>
#include <string>
#include <iostream>
#include <fstream>
#include <limits>
#include <algorithm>
int myMap::evalutionFunction(int x,int y){
    return    pow(posEnd.x-(posCurrent.x+x),2)+pow(posEnd.y-(posCurrent.y+y),2);
}

void myMap::init(std::string filename){
    std::ifstream istrm;
    istrm.open(filename,std::ios::in);
    istrm>>length>>width;
    istrm.get();
    for(int w=0;w<width;w++)
    {
        std::string st;
        getline(istrm,st);
        auto posS = st.find('s');
        if(posS!=st.npos){
            posStart.x = w;
            posStart.y = posS;
        }
        auto posE = st.find('e');
        if(posE!=st.npos){
            posEnd.x = w;
            posEnd.y = posE;
        }
        m.push_back(st);
    }
    posCurrent = posStart;

}
int myMap::findPath() {
    if(posCurrent==posEnd)
        return 0;
    std::vector<int> value{};
```

```
    int upDown[]{-1,1,0,0};
    int leftRight[]{0,0,-1,1};
    for(int i=0;i<4;i++)
    {
        int nx = posCurrent.x+upDown[i],ny = posCurrent.y+leftRight[i];
        if(nx>=0&&nx<length&&ny>=0&&ny<width&&m[nx][ny]!='*'&&m[nx][ny]!='p')
            value.push_back(evalutionFunction(upDown[i],leftRight[i]));
        else
            value.push_back(std::numeric_limits<int>::max());
    }

    std::vector<int>::iterator p = std::min_element(value.begin(),value.end());
    auto d = std::distance(value.begin(),p);
    posCurrent.x +=upDown[d];
    posCurrent.y +=leftRight[d];
    if(m[posCurrent.x][posCurrent.y]!='e')
        m[posCurrent.x][posCurrent.y] = 'p';
    findPath();
}
```

六 实验结果以及分析

```
gaolex@HASEE:/mnt/c/Users/gaolex/Desktop/tyut-ai-homework$ g++ main.cpp heuristicSearch.cpp -std=c++11
gaolex@HASEE:/mnt/c/Users/gaolex/Desktop/tyut-ai-homework$ ./a.out

**********
*s*    * *
*p*    * *
*p**   * *
*pp*     *
* p*     *
* p*     *
* p* pe  *
* pppp   *
gaolex@HASEE:/mnt/c/Users/gaolex/Desktop/tyut-ai-homework$
```

在迷宫问题的启发式搜索中，我使用可配置的地图文件来定义迷宫地图。s 代表入口、e 代表出口、*代表墙壁、p 代表从 s 到 e 走过的路线。每一步使用估计函数确定最优(估计函数值最小)路线，走过的路线的估计函数值标记为无穷大，防止在节点之间来回摇摆，陷入局部最优。

实验二 基于产生式系统的问题求解

一 实验目的

熟悉和掌握产生式系统的构成和运行机制，掌握基于规则推理的基本方法和技术。

二 实验内容以及原理

问题描述：用基于产生式系统的方法求解传教士和野人问题

有 N 个传教士和 N 个野人来到河边准备渡河，河岸有一条船，每次至多可供 K 个乘渡，问传教士为了安全起见，应如何规划摆渡方案，使得任何时刻，河岸两边以及船上的野人数目总是不超过传教士的数目，即求解传教士和野人从左岸全部摆渡到右岸的过程中，任何时刻满足 M（传教士数）≥C（野人数）和 M+C≤K 的摆渡方案。

产生式系统（Production system）首先由波斯特（Post）于 1943 年提出的产生式规则（Production rule）而得名，他们用这种规则对符号串进行置换运算，后来，美国的纽厄尔和西蒙利用这个原理建立了一个人类的认知模型（1965 年），同年，斯坦福大学利用产生式系统结构设计出第一个专家系统 DENDRAL。

产生式系统用来描述若干个不同的以一个基本概念为基础的系统。这个基本概念就是产生式规则或产生式条件和操作对象的概念。

在产生式系统中，论域的知识分为两部份：

1. 事实：用于表示静态知识，如事物、事件和它们之间的关系；
2. 规则：用于表示推理过程和行为

一个产生式系统由三个部分组成，如图所示：

1. 总数据库：用来存放与求解问题有关的数据以及推理过程环境的当前状态的描述。
2. 产生式规则库：主要存放问题求解中的规则。
3. 控制策略：其作用是说明下一步应该选用什么规则，也就是说如何应用规则。

三 实验仪器

Intel(R) Core(TM) i5-4210M CPU @ 2.6GHz 2.6GHz
8G RAM
Windows10 专业版 1803 Linux 子系统 Ubuntu 16.04
VS code +；Cmder；g++5.4

四 实验步骤/设计实现

1. 状态的定义：

(X Y : M N)

X = 左河岸野人的数量
Y = 左河岸传教士的数量
M = 右河岸野人的数量
N = 右河岸传教士的数量

^ = 船在河岸的哪边

< A B >

A = 野人渡河的数量

B = 传教士渡河的数量

初始状态：

(3 3 : 0 0)

^

目标状态：

(0 0 : 3 3)

^

2. 可能的动作

1.< 1 0 >

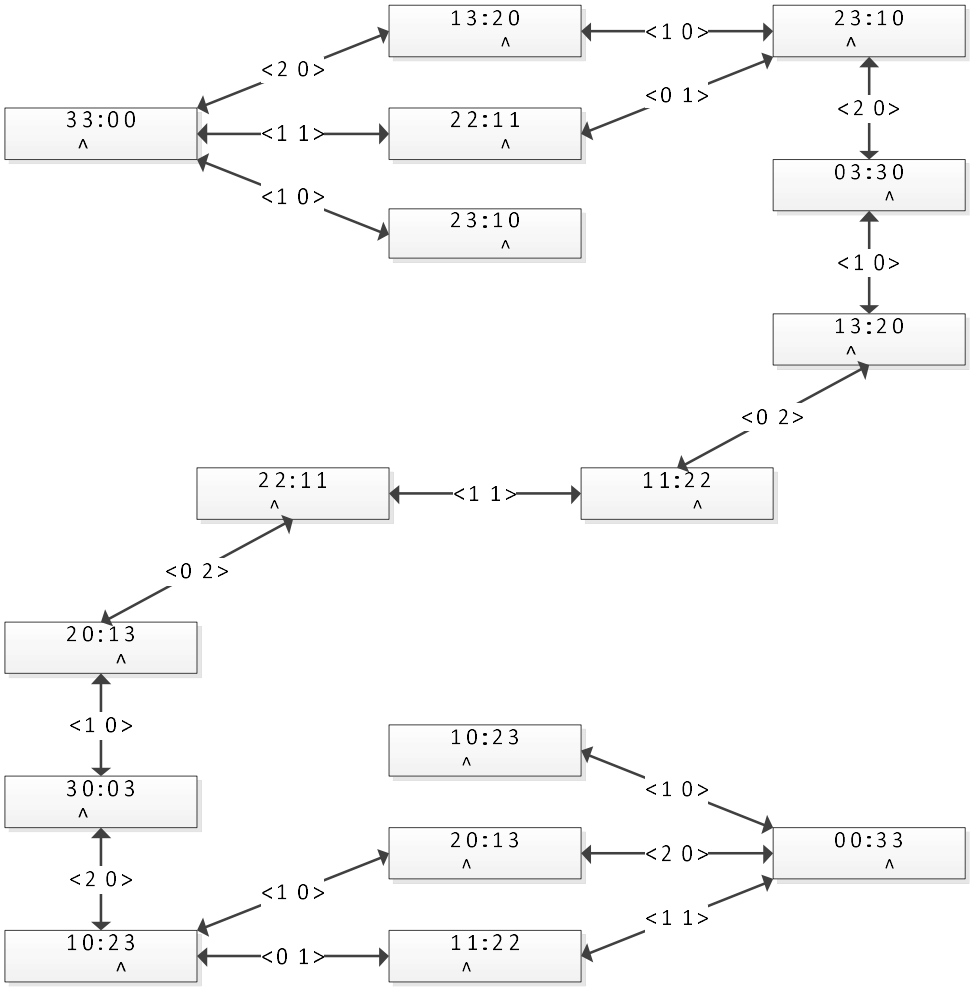
2.< 2 0 >

3.< 0 1 >

4.< 0 2 >

5.< 1 1 >

3. 状态解空间图：



五 实验代码

```
//cannibal.cpp
//Created by gaolex
#include<iostream>
#include "tree.h"
using namespace std;
int main()
{
   SearchTree MyTree;
   node* home = new node;
   home->parent = NULL;
   home->depth = 0;
   home->leafNum = 0;
   for(int i = 0; i < 5; i++)
      {      home->leaf[i]      =
NULL; }
   home->location = 'X';
   home->X.missionary = 3;
   home->X.cannibal = 3;
   home->Y.missionary = 0;
   home->Y.cannibal = 0;
   MyTree.travers(home);
   return 0;
}
```

```
//tree.h part1
#ifndef TREE_H_
#define TREE_H_
using namespace std;
struct land
{
   int missionary;
   int cannibal;
};
struct previousStates
{
   int Xmissionary;
   int Ymissionary;
   int Xcannibal;
   int Ycannibal;
   char location;
};

struct path
{
   char location;
   int depth;
   land X;
   land Y;
};
struct node
{
   char location;
   int depth;
   land X;
   land Y;
```

```
//tree.h part2
   node* leaf[5];
   node* parent;
   int leafNum;
};
class SearchTree
{
 private:
   int numberOfStates;
   previousStates
visitedStates[20];
   int pathLength;
   path fullPath[15];

 public:
   SearchTree();
   ~SearchTree();
   node*      createNode(node*
parent, int leafNum);
   node*     invalidNode(node*
current);
   int validate(node* validate);
   void feiry(node* feiry);
   node*   addValidState(node*
current);
   void
removeLastPathEntry();
   void travers(node* current);
   void display(node* current);
};
#endif
```

```
#include <iostream>
#include <string>
#include "tree.h"
using namespace std;
#define MAX_DEPTH 100
node *redFlag;

SearchTree::SearchTree()
{
```

```
    for(int i = 0; i < 20; i++)
        {
            visitedStates[i].Xmissionary = 0;
            visitedStates[i].Ymissionary = 0;
            visitedStates[i].Xcannibal = 0;
            visitedStates[i].Ycannibal = 0;
            visitedStates[i].location = 'Q';
        }
    numberOfStates = 0;
    pathLength = 15;
    while(pathLength != 0)
        { removeLastPathEntry(); }
}

SearchTree::~SearchTree()
{ ; }
node* SearchTree::createNode(node *parent, int leafNum)
{
    node *temp;
    temp = new node;
    temp->leafNum = leafNum;
    for(int i = 0; i < 5; i++)
        { temp->leaf[i] = NULL; }
    temp->parent = parent;
    temp->depth = parent->depth + 1;
    parent->leaf[leafNum] = temp;
    return temp;
}
node* SearchTree::invalidNode(node *current)
{
    int invalidLeaf = current->leafNum;
    current = current->parent;
    delete current->leaf[invalidLeaf];
    current->leaf[invalidLeaf] = redFlag;
    return current;
}
int SearchTree::validate(node *validate)
{
    int Xmis = validate->X.missionary;
    int Ymis = validate->Y.missionary;
    int Xcan = validate->X.cannibal;
    int Ycan = validate->Y.cannibal;
    if((Xcan > Xmis && Xmis !=0) || (Ycan > Ymis && Ymis !=0)
        || Xmis > 3 || Xmis < 0 || Ymis > 3 || Ymis < 0
```

```
        || Xcan > 3 || Xcan < 0 || Ycan > 3 || Ycan < 0)
      { return 0; }
   else if(Ycan == 3 && Ymis == 3)
      { return 2; }
   else
      {
        for(int i = 0; i < numberOfStates; i++)
      {
        if(visitedStates[i].Xmissionary == Xmis
            && visitedStates[i].Xcannibal == Xcan
           && visitedStates[i].location == validate->location)
            { return 0; }
      }
        return 1;
      }
}
void SearchTree::feiry(node *feiry)
{
   int numM, numC;
   numM = 0;
   numC = 0;
   switch (feiry->leafNum)
      {
      case 0:
        numM = 1;
        break;
      case 1:
        numM = 2;
        break;
      case 2:
        numC = 1;
        break;
      case 3:
        numC = 2;
        break;
      case 4:
        numM = 1;
        numC = 1;
        break;
      }
   if(feiry->depth % 2 != 0)
      {
        feiry->X.missionary -= numM;
        feiry->Y.missionary += numM;
```

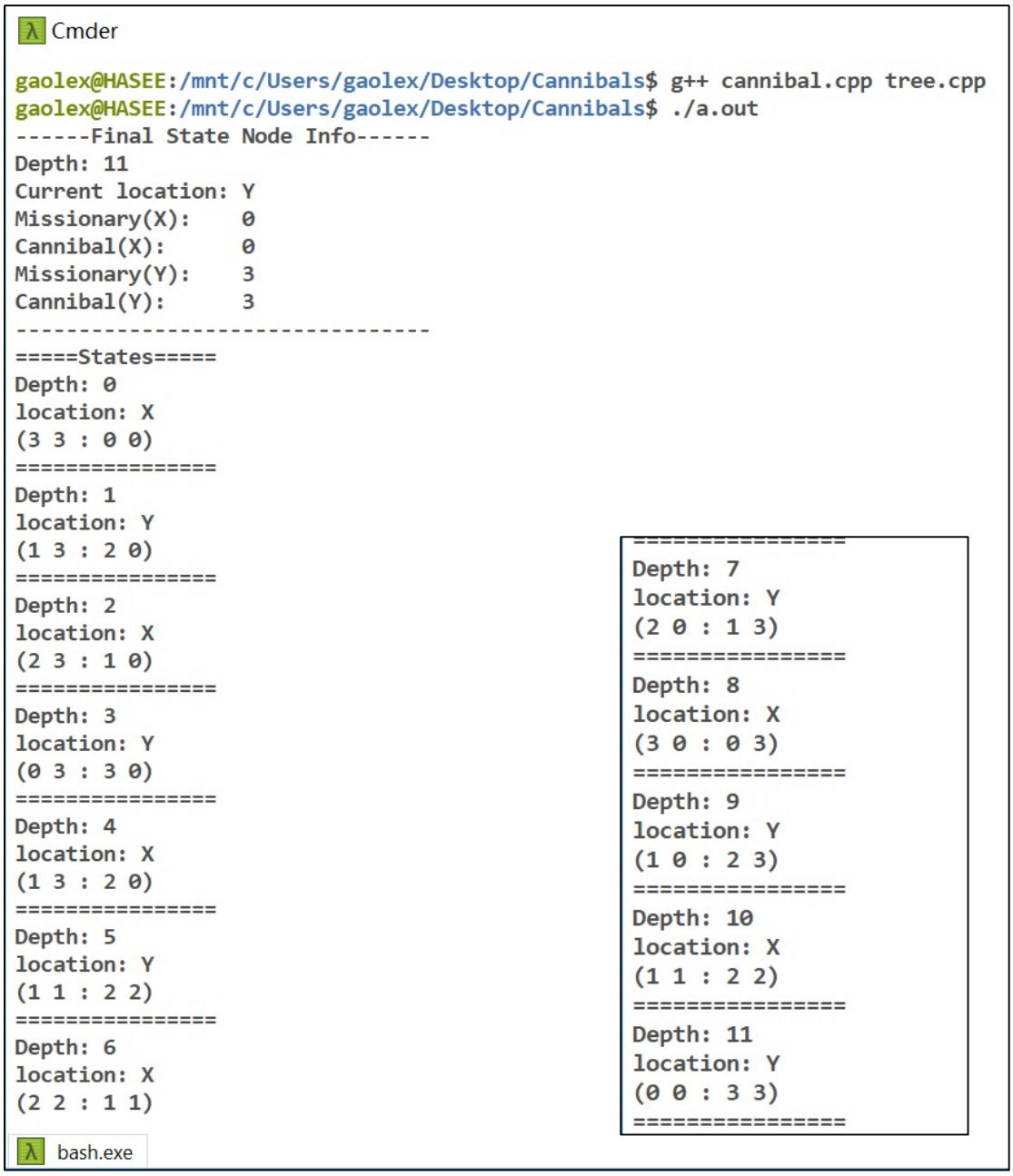
```
            feiry->X.cannibal -= numC;
            feiry->Y.cannibal += numC;
            feiry->location = 'Y';
        }
    else
        {
            feiry->X.missionary += numM;
            feiry->Y.missionary -= numM;
            feiry->X.cannibal += numC;
            feiry->Y.cannibal -= numC;
            feiry->location = 'X';
        }
}
void SearchTree::display(node *current)
{
    cout << "------Final State Node Info------\n"
            << "Depth: " << current->depth << "\n"
            << "Current location: " << current->location << "\n"
            << "Missionary(X):       " << current->X.missionary << "\n"
            << "Cannibal(X):         " << current->X.cannibal << "\n"
            << "Missionary(Y):       " << current->Y.missionary << "\n"
            << "Cannibal(Y):         " << current->Y.cannibal << "\n"
            << "---------------------------------\n";
}
node* SearchTree::addValidState(node *current)
{
    visitedStates[numberOfStates].Xmissionary = current->X.missionary;
    visitedStates[numberOfStates].Ymissionary = current->Y.missionary;
    visitedStates[numberOfStates].Xcannibal = current->X.cannibal;
    visitedStates[numberOfStates].Ycannibal = current->Y.cannibal;
    visitedStates[numberOfStates].location = current->location;
    numberOfStates++;
    fullPath[pathLength].location = current->location;
    fullPath[pathLength].depth = current->depth;
    fullPath[pathLength].X.missionary = current->X.missionary;
    fullPath[pathLength].Y.missionary = current->Y.missionary;
    fullPath[pathLength].X.cannibal = current->X.cannibal;
    fullPath[pathLength].Y.cannibal = current->Y.cannibal;
    pathLength++;
}
void SearchTree::removeLastPathEntry()
{
    pathLength--;
    fullPath[pathLength].location = 'Q';
```

```
    fullPath[pathLength].depth = -1;
    fullPath[pathLength].X.missionary = 0;
    fullPath[pathLength].X.cannibal = 0;
    fullPath[pathLength].Y.missionary = 0;
    fullPath[pathLength].Y.cannibal = 0;
}
void SearchTree::travers(node *current)
{
    int validMove = 0;
    node *temp;
    temp = new node;
    int tempLeafNum = 0;
    redFlag = new node;
    addValidState(current);
    while(current->depth < MAX_DEPTH)
        {
          for(int i = 0; i < 5; i++)
        {
          if(current->leaf[i] == NULL)
              {
                current = createNode(current, i);
                break;
              }
          else if(i == 4)
              {
                tempLeafNum = current->leafNum + 1;
                current = current->parent;
                current = createNode(current, tempLeafNum);
                removeLastPathEntry();
              }
        }
          if(current->depth > 0)
        {
          current->X.cannibal = current->parent->X.cannibal;
          current->Y.cannibal = current->parent->Y.cannibal;
          current->X.missionary = current->parent->X.missionary;
          current->Y.missionary = current->parent->Y.missionary;
        }
          feiry(current);
          validMove = validate(current);
          if(validMove == 1)
        {
          addValidState(current);
        }
```

```
        else if(validMove == 0)
    {
        current = invalidNode(current);
    }
        else if(validMove == 2)
    {
        addValidState(current);
        display(current);

        cout<<"=====States====="<<endl;
        for(int i = 0; i < pathLength; i++)
            {
                cout<<"Depth: "<<fullPath[i].depth<<endl;
                cout<<"location: "<<fullPath[i].location<<endl;
                cout<<"("<<fullPath[i].X.cannibal<<" "<<fullPath[i].X.missionary<<" : "
                <<fullPath[i].Y.cannibal<<" "<<fullPath[i].Y.missionary<<")"<<endl;
                cout<<"================"<<endl;
            }
        return;
    }
    }
}
```

六 实验结果以及分析



通过该实验，我理解了产生式系统推理的构成和运行机制，掌握了基于规则推理的基本方法与技术。

实验三 归结原理

一 实验目的

熟悉和掌握归结原理的基本思想和基本方法，通过实验培养了我们利用逻辑方法表示知识，并掌握采用机器推理来进行问题求解的基本方法。

二 实验内容以及原理

问题描述：四对夫妇中，王结婚时，周送了礼；周和钱是同一排球队的队员；李的爱人是陈的爱人的表哥；陈夫妇与邻居吵架时，徐、周、吴的爱人都去助战；李、徐、周结婚前住在同一宿舍，试用归结原理求王、周、钱、陈、李、徐、吴、孙几人谁和谁是夫妇。

归结是一种应用于谓词演算中的定理证明技术，从 60 年代中期开始，它就成为人工智能问题求解研究的一个组成部分。归结原理描述了如何用最少的合一次数在一个子句数据库中发现矛盾的方法。归 结否证定理证明的方法是，对所要证明的命题取反，把它加到一个已知为真的公里集中，然后用归结推理规则证明这将导致一个矛盾，一旦定理证明程序证明了否定目标与已知的公理集合不一致，就能推导出原来的目标与公理集是一致的。这就证明了该定理。

三 实验仪器

Intel(R) Core(TM) i5-4210M CPU @ 2.6GHz 2.6GHz
8G RAM
Windows10 专业版 1803 Linux 子系统 Ubuntu 16.04
VS code +；Cmder；g++5.4

四 实验步骤/设计实现

（1）把前提或公理转换成子句形式；
（2）把求证目标的否定的子句形式加到公理集合中；
（3）对所有这些子句进行归结，产生它们的逻辑结果子句；
（4）用产生空子句的方法来得出矛盾；
（5）否定目标的否证在用于产生空子句的代换下为真。

五 实验代码

```
import java.lang.*;

public class gaolex {
    public class Person {
        private String name;
        private boolean sex;
        private Person spouse;
```

```
    public Person(String name) {
        this.name = name;
        this.sex = false;
        this.spouse = null;
    }

    public Person() {
        name = null;
        sex = false;
        spouse = null;
    }

    public Person(String name, boolean sex, Person spouse) {
        this.name = name;
        this.sex = sex;
        this.spouse = spouse;
    }

    public void setName(String name) {
        this.name = name;
    }

    public String getName() {
        return name;
    }

    public void setSex(boolean sex) {
        this.sex = sex;
    }

    public boolean getSex() {
        return sex;
    }

    public void setSpouse(Person person1) {
        this.spouse = person1;
    }

    public Person getSpouse() {
        return spouse;
    }
}
```

```
Person[] sortBySpouse1 = new Person[8];
Person[] sortBySpouse2 = new Person[8];
int x = 0, y = 0;
Person[] sortByEqualSex1 = new Person[8];
Person[] sortByEqualSex2 = new Person[8];
int n = 0, p = 0, f = 0;
Person[] woman = new Person[4];
Person[] man = new Person[4];
int k = 0, s = 0, r = 0;
boolean sign = true;
boolean sign1 = true;
boolean sign2 = true;
boolean sign3 = true;
boolean sign4 = true;
boolean sign5 = true;
boolean sign6 = true;
boolean sign7 = true;
Person[] couple1 = new Person[4];
Person[] couple2 = new Person[4];
Person[] couple3 = new Person[4];
Person[] couple4 = new Person[4];
int a = 0, b = 0, c = 0, d = 0;
int a2 = 0, b2 = 0, c2 = 0, d2 = 0;

public void isCouple(Person person1, Person person2) {
    if (person1.getSpouse().getName() == person2.getName()) {
        person1.setSex(!person2.getSex());
        person2.setSex(!person1.getSex());
        System.out.println(person1.getName() + "的性别和"
                + person2.getName() + "的性别不相同");
    }
}

public void marriage(Person person1, boolean person1Marriage,
        Person person2, boolean person2giveGiftPerson1) {
    if (person1Marriage && person2giveGiftPerson1)
        if (!person1.getName().equals(person2.getName()))
            System.out.println(person1.getName() + "  和  "
                    + person2.getName() + "  不是夫妻");
    sortBySpouse1[x++] = person1;
    sortBySpouse2[y++] = person2;
}

public void oneVolleyballGroup(boolean oneGroup, Person person1,
```

```
            Person person2) {
        if (oneGroup && person1.getSex() == person2.getSex())
            System.out.println(person1.getName() + "  和  " + person2.getName()
                    + "  是相同的性别");
        sortByEqualSex1[n++] = person1;
        sortByEqualSex1[n++] = person2;
        f = n;
    }

    public void liveCollectivityDorm(boolean liveTogether, Person person1,
            Person person2, Person person3) {
        if (liveTogether && !person1.getName().equals(person2.getName())
                && !person1.getName().equals(person3.getName())
                && !person2.getName().equals(person3.getName())) {
            System.out.println(person1.getName() + "的性别和  "
                    + person2.getName() + "的性别相同  "
                    + person3.getName() + "的性别相同");
            sortByEqualSex2[p++] = person1;
            sortByEqualSex2[p++] = person2;
            sortByEqualSex2[p++] = person3;
        }
    }

    public void isbrotherInLaw(Person person1, Person person2,
            boolean p1SIsBrotherInLawP2S) {
        if (!person1.getName().equals(person2.getName())
                && p1SIsBrotherInLawP2S) {
            person1.setSex(false);
            sortBySpouse1[x++] = person1;
            sortBySpouse2[y++] = person2;
            System.out.println(person1.getName() + "  是一个女人");
            woman[k++] = person1;
            r = k;
        }
        System.out.println(person1.getName() + "的性别是女和  "
                + person1.getName() + "的配偶是男");
    }

    public void assistInFighting(boolean cCoupleQuarrel, Person cCouple,
            Person person1, Person person2, Person person3) {
        if (cCoupleQuarrel)
            if (!cCouple.getName().equals(person1.getName())
                    && !cCouple.getName().equals(person2.getName())
                    && !cCouple.getName().equals(person3.getName())
```

```
                    && !person1.getName().equals(person2.getName())
                    && !person1.getName().equals(person3.getName())
                    && !person2.getName().equals(person3.getName())) {
                System.out.println(cCouple.getName() + "的配偶不是"
                        + person1.getName() + " 和 " + cCouple.getName()
                        + "的配偶不是 " + person1.getName()
                        + " 的配偶");
                System.out.println(cCouple.getName() + "的配偶不是 "
                        + person2.getName() + " 和 " + cCouple.getName()
                        + "的配偶不是 " + person2.getName()
                        + " 的配偶");
                System.out.println(cCouple.getName() + "的配偶不是 "
                        + person3.getName() + " 和 "  + cCouple.getName()
                        + "的配偶不是 " + person3.getName()
                        + " 的配偶");
            }
        sortBySpouse1[x++] = cCouple;
        sortBySpouse2[y++] = person1;
        sortBySpouse1[x++] = cCouple;
        sortBySpouse2[y++] = person2;
        sortBySpouse1[x++] = cCouple;
        sortBySpouse2[y++] = person3;
        sortBySpouse1[x++] = person1;
        sortBySpouse2[y++] = person2;
        sortBySpouse1[x++] = person1;
        sortBySpouse2[y++] = person3;
        sortBySpouse1[x++] = person2;
        sortBySpouse2[y++] = person3;
    }

    public void gaolexObjByRules() {
        Person[] person = new Person[8];
        person[0] = new Person("王", false, person[0]);
        person[1] = new Person("陈", false, person[1]);
        person[2] = new Person("周", false, person[2]);
        person[3] = new Person("钱", false, person[3]);
        person[4] = new Person("吴", false, person[4]);
        person[5] = new Person("孙", false, person[5]);
        person[6] = new Person("李", false, person[6]);
        person[7] = new Person("许", false, person[7]);
        marriage(person[0], true, person[2], true);
        oneVolleyballGroup(true, person[2], person[3]);
        isbrotherInLaw(person[6], person[1], true);
        assistInFighting(true, person[1], person[7], person[2], person[4]);
```

```
liveCollectivityDorm(true, person[6], person[7], person[2]);
for (int v = 0; v < f; v++)
    for (int t = 0; t < p; t++) {
        if (sortByEqualSex1[v].getName().equals(
                sortByEqualSex2[t].getName())) {
            for (int z = 0; z < p; z++) {
                for (int j = 0; j < f; j++) {
                    if (!sortByEqualSex2[z].getName().equals(
                            sortByEqualSex1[j].getName()))
                        ;
                    else {
                        sortByEqualSex1[j] = sortByEqualSex2[z];
                        sign2 = false;
                    }
                }
                if (sign2)
                    sortByEqualSex1[n++] = sortByEqualSex2[z];
                sign2 = true;
            }
        }
    }
for (int m = 0; m < r; m++)
    for (int q = 0; q < n; q++) {
        if (woman[m].getName().equals(sortByEqualSex1[q].getName())) {
            for (int l = 0; l < n; l++) {
                for (int j = 0; j < r; j++) {
                    if (!sortByEqualSex1[l].getName().equals(
                            woman[j].getName()))
                        ;
                    else {
                        woman[j] = sortByEqualSex1[l];
                        sign = false;
                    }
                }
                if (sign)
                    woman[k++] = sortByEqualSex1[l];
                sign = true;
            }
        }
    }
r = k;
if (r == 4) {
    for (int j = 0; j < 8; j++) {
        int w = 0;
```

```
                    for (int i = 0; i < r; i++) {
                        if (!person[j].getName().equals(woman[i].getName()))
                            w++;
                    }
                    if (w == r)
                        man[s++] = person[j];
                }
            }
            System.out.println();
            System.out.println("得到的男和女分别是：");
            System.out.print("女  :" + "        ");
            for (int i = 0; i < r; i++) {
                woman[i].setSex(false);
                System.out.print(woman[i].getName() + "          ");
            }
            System.out.println();
            System.out.println();
            System.out.print("男      :" + "        ");
            for (int j = 0; j < s; j++) {
                man[j].setSex(true);
                System.out.print(man[j].getName() + "          ");
            }
            System.out.println();
            System.out.println();
            for (int i = 0; i < x; i++)
                if (sortBySpouse1[i].getSex() != sortBySpouse2[i].getSex()) {
                    if (!sortBySpouse1[i].getSex()) {
                        Person temp = new Person();
                        temp = sortBySpouse1[i];
                        sortBySpouse1[i] = sortBySpouse2[i];
                        sortBySpouse2[i] = temp;
                    }
                }
        }

        public void partitionCouple() {
            for (int i1 = 0; i1 < x; i1++)
                for (int i2 = 0; i2 < s; i2++) {
                    if (sortBySpouse1[i1].getName().equals(man[i2].getName())) {
                        for (int i = 0; i < r; i++) {
                            if (!sortBySpouse2[i1].getName().equals(
                                    woman[i].getName())
                                    && sortBySpouse2[i1].getSex() == woman[i]
                                            .getSex())
```

```
                if (man[i2].getName().equals("王")) {
                    int a1 = a;
                    for (int j = 0; j < a1; j++) {
                        if (!couple1[j].getName().equals(
                                woman[i].getName()))
                            ;
                        else {
                            couple1[j] = woman[i];
                            sign3 = false;
                        }
                    }
                    if (sign3)
                        couple1[a++] = woman[i];
                    sign3 = true;
                } else if (man[i2].getName().equals("陈")) {
                    int b1 = b;
                    for (int j = 0; j < b1; j++) {
                        if (!couple2[j].getName().equals(
                                woman[i].getName()))
                            ;
                        else {
                            couple2[j] = woman[i];
                            sign4 = false;
                        }
                    }
                    if (sign4)
                        couple2[b++] = woman[i];
                    sign4 = true;
                } else if (man[i2].getName().equals("吴")) {
                    int c1 = c;
                    for (int j = 0; j < c1; j++)
                        if (!couple3[j].getName().equals(
                                woman[i].getName()))
                            ;
                        else {
                            couple3[j] = woman[i];
                            sign5 = false;
                        }
                    if (sign5)
                        couple3[c++] = woman[i];
                    sign5 = true;
                } else {
                    int d1 = d;
                    for (int j = 0; j < d; j++)
```

```
                                        if (!couple4[j].getName().equals(
                                                woman[i].getName()))
                                            ;
                                        else {
                                            couple4[j] = woman[i];
                                            sign6 = false;
                                        }
                                    if (sign6)
                                        couple4[d++] = woman[i];
                                    sign6 = true;
                                }
                            }
                        }
                    }
                for (int i = 0; i < x; i++) {
                    if (sortBySpouse1[i].getName().equals("王"))
                        for (int j = 0; j < a; j++)
                            if (sortBySpouse2[i].getName().equals(couple1[j].getName())) {
                                if (j == a - 1)
                                    a--;
                                else {
                                    for (int j1 = j; j1 < a - 1; j1++)
                                        couple1[j1] = couple1[j1 + 1];
                                    a--;
                                }
                            }
                }
                for (int i = 0; i < x; i++) {
                    if (sortBySpouse1[i].getName().equals("陈"))
                        for (int j = 0; j < b; j++)
                            if (sortBySpouse2[i].getName().equals(couple2[j].getName())) {
                                if (j == b - 1)
                                    b--;
                                else {
                                    for (int j1 = j; j1 < b - 1; j1++)
                                        couple2[j1] = couple2[j1 + 1];
                                    b--;
                                }
                            }
                }
                for (int i = 0; i < x; i++) {
                    if (sortBySpouse1[i].getName().equals("吴"))
                        for (int j = 0; j < c; j++)
                            if (sortBySpouse2[i].getName().equals(couple3[j].getName())) {
```

```
                        if (j == c - 1)
                            c--;
                        else {
                            for (int j1 = j; j1 < c - 1; j1++)
                                couple3[j1] = couple3[j1 + 1];
                            c--;
                        }
                    }
            }
            for (int i = 0; i < x; i++) {
                if (sortBySpouse1[i].getName().equals("孙"))
                    for (int j = 0; j < d; j++)
                        if (sortBySpouse2[i].getName().equals(couple4[j].getName())) {
                            if (j == d - 1)
                                d--;
                            else {
                                for (int j1 = j; j1 < d - 1; j1++)
                                    couple4[j1] = couple4[j1 + 1];
                                d--;
                            }
                        }
            }
            if (a == 1)
                man[0].setSpouse(couple1[a - 1]);
            else
                for (int i = 0; i < a; i++) {
                    if (b == 1)
                        if (couple1[i].getName().equals(couple2[0].getName())) {
                            if (i == a - 1)
                                a--;
                            else {
                                for (int j1 = i; j1 < a - 1; j1++)
                                    couple1[j1] = couple1[j1 + 1];
                                a--;
                            }
                        }
                    if (c == 1)
                        if (couple1[i].getName().equals(couple3[0].getName())) {
                            if (i == a - 1)
                                a--;
                            else {
                                for (int j1 = i; j1 < a - 1; j1++)
                                    couple1[j1] = couple1[j1 + 1];
                                a--;
```

```
                        }
                    }
                if (d == 1)
                    if (couple1[i].getName().equals(couple4[0].getName())) {
                        if (i == a - 1)
                            a--;
                        else {
                            for (int j1 = i; j1 < a - 1; j1++)
                                couple1[j1] = couple1[j1 + 1];
                            a--;
                        }
                    }
            }
        if (b == 1)
            man[1].setSpouse(couple1[b - 1]);
        else
            for (int i = 0; i < b; i++) {
                if (a == 1)
                    if (couple2[i].getName().equals(couple1[0].getName())) {
                        if (i == b - 1)
                            b--;
                        else {
                            for (int j1 = i; j1 < b - 1; j1++)
                                couple2[j1] = couple2[j1 + 1];
                            b--;
                        }
                    }
                if (c == 1)
                    if (couple2[i].getName().equals(couple3[0].getName())) {
                        if (i == b - 1)
                            b--;
                        else {
                            for (int j1 = i; j1 < b - 1; j1++)
                                couple2[j1] = couple2[j1 + 1];
                            b--;
                        }
                    }
                if (d == 1)
                    if (couple2[i].getName().equals(couple4[0].getName())) {
                        if (i == d - 1)
                            d--;
                        else {
                            for (int j1 = i; j1 < b - 1; j1++)
                                couple2[j1] = couple2[j1 + 1];
```

```
                                b--;
                            }
                        }
                }
        if (c == 1)
                man[2].setSpouse(couple1[c - 1]);
        else
                for (int i = 0; i < c; i++) {
                        if (a == 1)
                                if (couple3[i].getName().equals(couple1[0].getName())) {
                                        if (i == c - 1)
                                                c--;
                                        else {
                                                for (int j1 = i; j1 < c - 1; j1++)
                                                        couple3[j1] = couple3[j1 + 1];
                                                c--;
                                        }
                                }
                        if (b == 1)
                                if (couple3[i].getName().equals(couple2[0].getName())) {
                                        if (i == c - 1)
                                                c--;
                                        else {
                                                for (int j1 = i; j1 < c - 1; j1++)
                                                        couple3[j1] = couple3[j1 + 1];
                                                c--;
                                        }
                                }
                        if (d == 1)
                                if (couple3[i].getName().equals(couple4[0].getName())) {
                                        if (i == c - 1)
                                                c--;
                                        else {
                                                for (int j1 = i; j1 < c - 1; j1++)
                                                        couple3[j1] = couple3[j1 + 1];
                                                c--;
                                        }
                                }
                }
        if (d == 1)
                man[3].setSpouse(couple1[d - 1]);
        else
                for (int i = 0; i < d; i++) {
                        if (a == 1)
```

```
                        if (couple4[i].getName().equals(couple1[0].getName())) {
                            if (i == d - 1)
                                d--;
                            else {
                                for (int j1 = i; j1 < d - 1; j1++)
                                    couple4[j1] = couple4[j1 + 1];
                                d--;
                            }
                        }
                    if (b == 1)
                        if (couple4[i].getName().equals(couple2[0].getName())) {
                            if (i == d - 1)
                                d--;
                            else {
                                for (int j1 = i; j1 < d - 1; j1++)
                                    couple4[j1] = couple4[j1 + 1];
                                d--;
                            }
                        }
                    if (d == 1)
                        if (couple4[i].getName().equals(couple4[0].getName())) {
                            if (i == d - 1)
                                d--;
                            else {
                                for (int j1 = i; j1 < d - 1; j1++)
                                    couple4[j1] = couple4[j1 + 1];
                                d--;
                            }
                        }
                }
    }

    public void supposeJudge() {
        if (a == 1) {
            man[0].setSpouse(couple1[0]);
            a2 = a;
        }
        if (b == 1) {
            man[1].setSpouse(couple2[0]);
            b2 = b;
        }
        if (c == 1) {
            man[2].setSpouse(couple3[0]);
            c2 = c;
```

```
        }
        if (d == 1) {
            man[3].setSpouse(couple4[0]);
            d2 = d;
        }
        if (a > 1) {
            man[0].setSpouse(couple1[0]);
            if (b == 0 && a >= 2) {
                man[1].setSpouse(couple1[1]);
                b++;
                if (c == 0 && a >= 3) {
                    {
                        man[2].setSpouse(couple1[2]);
                        couple3[c] = couple1[2];
                        c++;
                    }
                    if (d == 0 && a > 3) {
                        man[3].setSpouse(couple1[3]);
                        couple4[d] = couple1[3];
                        d++;
                    }
                }
            } else if (b != 0 && c == 0 && a >= 2) {
                {
                    man[2].setSpouse(couple1[1]);
                    couple3[c] = couple1[1];
                    c++;
                }
                if (d == 0 && a > 2) {
                    man[3].setSpouse(couple1[2]);
                    couple4[d] = couple1[2];
                    d++;
                }
            } else if (b != 0 && c != 0 && d == 0) {
                man[3].setSpouse(couple1[1]);
                couple4[d] = couple1[1];
                d++;
            }
            if (b == 1 && c == 0 && d == 0) {
                for (int i = 0; i < r; i++)
                    if (!woman[i].getName().equals(couple1[0].getName())
                            && !woman[i].getName().equals(couple1[1].getName()))
                        if (sign7 == true) {
                            couple3[0] = woman[i];
```

```
                                man[2].setSpouse(couple3[0]);
                                sign7 = false;
                            } else {
                                couple4[0] = woman[i];
                                man[3].setSpouse(couple4[0]);
                                sign7 = true;
                            }
                    }
                    if (b == 1 && c == 1 && d == 0) {
                        for (int i = 0; i < r; i++)
                            if (!woman[i].getName().equals(couple1[0].getName())
                                    && !woman[i].getName().equals(couple1[1].getName())
                                    && !woman[i].getName().equals(couple2[0].getName())) {
                                couple4[0] = woman[i];
                                man[3].setSpouse(couple4[0]);
                            }
                    }
                }
                for (int j = 0; j < x; j++) {
                    if (man[0].getName().equals(sortBySpouse1[j].getName())) {
                        if (man[0].getSpouse().getName().equals(
                                sortBySpouse2[j].getName())) {
                            couple1[0] = man[0];
                            man[0].setSpouse(couple1[0]);
                        } else if (b2 == 1
                                && man[0].getSpouse().getName().equals(
                                        couple2[0].getName())) {
                            couple1[0] = man[0];
                            man[0].setSpouse(couple1[0]);
                        } else if (c2 == 1
                                && man[0].getSpouse().getName().equals(
                                        couple3[0].getName())) {
                            couple1[0] = man[0];
                            man[0].setSpouse(couple1[0]);
                        } else if (d2 == 1
                                && man[0].getSpouse().getName().equals(
                                        couple4[0].getName())) {
                            couple1[0] = man[0];
                            man[0].setSpouse(couple1[0]);
                            System.out.println(couple1[0].getName());
                        }
                    }
                }
                for (int j = 0; j < x; j++) {
```

```
            if (man[1].getName().equals(sortBySpouse1[j].getName())) {
                if (man[1].getSpouse().getName().equals(
                        sortBySpouse2[j].getName())) {
                    couple2[0] = man[1];
                    man[1].setSpouse(couple2[0]);
                } else if (a2 == 1
                        && man[1].getSpouse().getName().equals(
                                couple1[0].getName())) {
                    couple2[0] = man[1];
                    man[1].setSpouse(couple2[0]);
                } else if (c2 == 1
                        && man[1].getSpouse().getName().equals(
                                couple3[0].getName())) {
                    couple2[0] = man[1];
                    man[1].setSpouse(couple2[0]);
                } else if (d2 == 1
                        && man[1].getSpouse().getName().equals(
                                couple4[0].getName())) {
                    couple2[0] = man[1];
                    man[1].setSpouse(couple2[0]);
                }
            }
        }
        for (int j = 0; j < x; j++) {
            if (man[2].getName().equals(sortBySpouse1[j].getName()))
                if (man[2].getSpouse().getName().equals(
                        sortBySpouse2[j].getName())) {
                    couple3[0] = man[2];
                    man[2].setSpouse(couple3[0]);
                } else if (b2 == 1
                        && man[2].getSpouse().getName().equals(
                                couple2[0].getName())) {
                    couple3[0] = man[2];
                    man[2].setSpouse(couple3[0]);
                } else if (a2 == 1
                        && man[2].getSpouse().getName().equals(
                                couple1[0].getName())) {
                    couple3[0] = man[2];
                    man[2].setSpouse(couple3[0]);
                } else if (d2 == 1
                        && man[2].getSpouse().getName().equals(
                                couple4[0].getName())) {
                    couple3[0] = man[2];
                    man[2].setSpouse(couple3[0]);
```

```
                }
            }
            for (int j = 0; j < x; j++) {
                if (man[3].getName().equals(sortBySpouse1[j].getName()))
                    if (man[3].getSpouse().getName().equals(
                            sortBySpouse2[j].getName())) {
                        couple4[0] = man[3];
                        man[3].setSpouse(couple4[0]);
                    } else if (b2 == 1
                            && man[3].getSpouse().getName().equals(
                                    couple2[0].getName())) {
                        couple4[0] = man[3];
                        man[3].setSpouse(couple4[0]);
                    } else if (c2 == 1
                            && man[3].getSpouse().getName().equals(
                                    couple3[0].getName())) {
                        couple4[0] = man[3];
                        man[3].setSpouse(couple4[0]);
                    } else if (a2 == 1
                            && man[3].getSpouse().getName().equals(
                                    couple1[0].getName())) {
                        couple4[0] = man[3];
                        man[3].setSpouse(couple4[0]);
                    }
            }
            for (int j1 = 0; j1 < x; j1++)
                if (man[0].getName().equals(sortBySpouse1[j1].getName())) {
                    if (man[0].getSpouse().getSex())
                        for (int j = 0; j < r; j++)
                            if (b2 == 1 && c2 == 1 && d2 == 1) {
                                if (!woman[j].getName().equals(
                                        sortBySpouse2[j1].getName())
                                        && !woman[j].getName().equals(
                                                couple2[0].getName())
                                        && !woman[j].getName().equals(
                                                couple3[0].getName())
                                        && !woman[j].getName().equals(
                                                couple4[0].getName())) {
                                    couple1[0] = woman[j];
                                    man[0].setSpouse(couple1[0]);
                                }
                            } else if (b2 == 1 && c2 == 1) {
                                if (!woman[j].getName().equals(
                                        sortBySpouse2[j1].getName())
```

```
                                        && !woman[j].getName().equals(
                                                couple2[0].getName())
                                        && !woman[j].getName().equals(
                                                couple3[0].getName())) {
                                    couple1[0] = woman[j];
                                    man[0].setSpouse(couple1[0]);
                                }
                            } else if (b2 == 1) {
                                if (!woman[j].getName().equals(
                                        sortBySpouse2[j1].getName())
                                        && !woman[j].getName().equals(
                                                couple2[0].getName())) {
                                    couple1[0] = woman[j];
                                    man[0].setSpouse(couple1[0]);
                                }
                            }
                        }
                    for (int j1 = 0; j1 < x; j1++)
                        if (man[1].getName().equals(sortBySpouse1[j1].getName())) {
                            if (man[1].getSpouse().getSex())
                                for (int j = 0; j < r; j++)
                                    if (a2 == 1 && c2 == 1 && d2 == 1) {
                                        if (!woman[j].getName().equals(
                                                sortBySpouse2[j1].getName())
                                                && !woman[j].getName().equals(
                                                        couple1[0].getName())
                                                && !woman[j].getName().equals(
                                                        couple3[0].getName())
                                                && !woman[j].getName().equals(
                                                        couple4[0].getName())) {
                                            couple2[0] = woman[j];
                                            man[1].setSpouse(couple2[0]);
                                        }
                                    } else if (a2 == 1 && c2 == 1) {
                                        if (!woman[j].getName().equals(
                                                sortBySpouse2[j1].getName())
                                                && !woman[j].getName().equals(
                                                        couple1[0].getName())
                                                && !woman[j].getName().equals(
                                                        couple3[0].getName())) {
                                            couple2[0] = woman[j];
                                            man[1].setSpouse(couple2[0]);
                                        }
                                    } else if (a2 == 1) {
```

```
                                if (!woman[j].getName().equals(
                                        sortBySpouse2[j1].getName())
                                        && !woman[j].getName().equals(
                                                couple1[0].getName())) {
                                    couple2[0] = woman[j];
                                    man[1].setSpouse(couple2[0]);
                                }
                            }
                    }
                for (int j1 = 0; j1 < x; j1++)
                    if (man[2].getName().equals(sortBySpouse1[j1].getName())) {
                        if (man[2].getSpouse().getSex())
                            for (int j = 0; j < r; j++)
                                if (a2 == 1 && b2 == 1 && d2 == 1) {
                                    if (!woman[j].getName().equals(
                                            sortBySpouse2[j1].getName())
                                            && !woman[j].getName().equals(
                                                    couple1[0].getName())
                                            && !woman[j].getName().equals(
                                                    couple2[0].getName())
                                            && !woman[j].getName().equals(
                                                    couple4[0].getName())) {
                                        couple3[0] = woman[j];
                                        man[2].setSpouse(couple3[0]);
                                    }
                                } else if (a2 == 1 && b2 == 1) {
                                    if (!woman[j].getName().equals(
                                            sortBySpouse2[j1].getName())
                                            && !woman[j].getName().equals(
                                                    couple1[0].getName())
                                            && !woman[j].getName().equals(
                                                    couple2[0].getName())) {
                                        couple3[0] = woman[j];
                                        man[2].setSpouse(couple3[0]);
                                    }
                                } else if (a2 == 1) {
                                    if (!woman[j].getName().equals(
                                            sortBySpouse2[j1].getName())
                                            && !woman[j].getName().equals(
                                                    couple1[0].getName())) {
                                        couple3[0] = woman[j];
                                        man[2].setSpouse(couple3[0]);
                                    }
                                }
```

```
            }
        for (int j1 = 0; j1 < x; j1++)
            if (man[3].getName().equals(sortBySpouse1[j1].getName())) {
                if (man[3].getSpouse().getSex())
                    for (int j = 0; j < r; j++)
                        if (a2 == 1 && b2 == 1 && c2 == 1) {
                            if (!woman[j].getName().equals(
                                    sortBySpouse2[j1].getName())
                                    && !woman[j].getName().equals(
                                            couple1[0].getName())
                                    && !woman[j].getName().equals(
                                            couple2[0].getName())
                                    && !woman[j].getName().equals(
                                            couple3[0].getName())) {
                                couple4[0] = woman[j];
                                man[3].setSpouse(couple4[0]);
                            }
                        } else if (a2 == 1 && b2 == 1) {
                            if (!woman[j].getName().equals(
                                    sortBySpouse2[j1].getName())
                                    && !woman[j].getName().equals(
                                            couple1[0].getName())
                                    && !woman[j].getName().equals(
                                            couple2[0].getName())) {
                                couple4[0] = woman[j];
                                man[3].setSpouse(couple4[0]);
                            }
                        } else if (a2 == 1) {
                            if (!woman[j].getName().equals(
                                    sortBySpouse2[j1].getName())
                                    && !woman[j].getName().equals(
                                            couple1[0].getName())) {
                                couple4[0] = woman[j];
                                man[3].setSpouse(couple4[0]);
                            }
                        }
            }
        for (int i = 0; i < r; i++)
            if (!woman[i].getName().equals(couple1[0].getName())
                    && !woman[i].getName().equals(couple2[0].getName())
                    && !woman[i].getName().equals(couple3[0].getName())) {
                couple4[0] = woman[i];
                man[3].setSpouse(couple4[0]);
            }
```

```
            for (int i = 0; i < r; i++)
                if (woman[i].getName().equals(couple1[0].getName()))
                    woman[i].setSpouse(man[0]);
                else if (woman[i].getName().equals(couple2[0].getName()))
                    woman[i].setSpouse(man[1]);
                else if (woman[i].getName().equals(couple3[0].getName()))
                    woman[i].setSpouse(man[2]);
                else if (woman[i].getName().equals(couple4[0].getName()))
                    woman[i].setSpouse(man[3]);
            System.out.println("得到的夫妻是：:");
            System.out.println("男      ---  女");
            for (int i = 0; i < r; i++)
                System.out.println(man[i].getName() + "                    "
                        + man[i].getSpouse().getName());
        }

        public static void main(String[] args) {
            gaolex gaolexObj = new gaolex();
            gaolexObj.gaolexObjByRules();
            gaolexObj.partitionCouple();
            gaolexObj.supposeJudge();
        }
}
```

六 实验结果以及分析

```
λ Cmder

gaolex@HASEE:/mnt/c/Users/gaolex/Desktop/tyut-ai-homework$ javac gaolex.java
gaolex@HASEE:/mnt/c/Users/gaolex/Desktop/tyut-ai-homework$ java gaolex
王 和 周 不是夫妻
周 和 钱 是相同的性别
李 是一个女人
李的性别是女和 李的配偶是男
陈的配偶不是许 和 陈的配偶不是 许 的配偶
陈的配偶不是 周 和 陈的配偶不是 周 的配偶
陈的配偶不是 吴 和 陈的配偶不是 吴 的配偶
李的性别和 许的性别相同 周的性别相同

得到的男和女分别是:
女 :    李      周      钱      许

男   :    王      陈      吴      孙

得到的夫妻是: :
男    --- 女
王            许
陈            钱
吴            李
孙            周
gaolex@HASEE:/mnt/c/Users/gaolex/Desktop/tyut-ai-homework$
```

通过本次实验掌握了一阶谓词逻辑和产生式表示这两种知识表示方法，同时也深入理解了产生式系统的推理机制。

实验四 简单遗传算法

一 实验目的

熟悉和掌握遗传算法的基本思想和基本方法，通过实验培养学生利用遗传算法进行问题求解的基本技能，并且了解进化计算其他分支的基本思想和基本方法。

二 实验内容以及原理

用遗传算法求解函数 f(x)=x*sin(10л+x)+1.0 在区间[-1，2]的最大值。

三 实验仪器

Intel(R) Core(TM) i5-4210M CPU @ 2.6GHz 2.6GHz
8G RAM
Windows10 专业版 1803 Linux 子系统 Ubuntu 16.04
VS code +；Cmder；g++5.4

四 实验步骤/设计实现

简单遗传算法的基本流程如下：
(1)初始化群体：随机产生一个由确定长度的特征串组成的初始群体
(2)计算群体上每个个体的适应度值
(3)按由个体适应度值所决定的规则选择将进入下一代的个体，也就是选择算子操作。
(4)按概率 Pc 对配对池中的个体进行交叉算子操作。
(5)按概率 Pm 对交叉算子产生的所有新个体进行变异操作。
(6)若没有满足某种停止条件，则转(2)，否则进入下一步。
(7)输出群体中适应度值最优的染色体作为问题的满意解或最优解。

五 实验代码

```
//ga.cpp
//created by gaolex
# include <stdio.h>
# include <stdlib.h>
# include <time.h>
# include <math.h>
/****** the definition of constant******/
# define PI 3.14159
# define POPSIZE 80
/****** the definition of user data*****/
# define LEFT -1
# define RIGHT 2
```

```
# define CHROMLENGTH 22

# define random(x) rand()%x

const int MaxGeneration=200;
const double Pc=0.6;
const double Pm=0.001;

/***** the definition of data structure*****/
struct individual
{
char chrom[CHROMLENGTH+1];//基因
double x;     //自变量
double value;//目标函数值
double fitness;//适应度
};

/***** the definition of global variables*****/
int generation;
int best_index;
int worst_index;
struct individual bestindividual;        //局部最优个体
struct individual worstindividual;       //局部最差个体
struct individual currentbest;           //全局最优个体
struct individual population[POPSIZE];//种群

/*****declaration of prototype  原型声明*****/
void GenerateInitialPopulation (void);       //初始化种群
void GenerateNextPopulation (void);          //产生下一代种群
void EvaluatePopulation (void);              //评估
long DecodeChromosome (char *,int,int);      //对基因进行解码
void CalculateObjectValue (void);            //计算目标函数值
void CalculateFitnessValue (void);           //计算适应值
void FindBestAndWorstIndividual (void);      //寻找最优及最差个体
void PerformEvolution (void);                //进化
void SelectionOperator (void);               //选择
void CrossoverOperator (void);               //交叉
void MutationOperator (void);                //变异
void OutputTextReport (void);



/***** main program*****/
int main (void)
{
```

```
    generation=0;
    GenerateInitialPopulation ();          //调用初始群体函数
    EvaluatePopulation ();                 //第一次评估
    while (generation<MaxGeneration)       //迭代一定代数
    {
        generation++;
        GenerateNextPopulation ();         //根据评估的结果来产生下一代
        EvaluatePopulation ();             //对新一代种群进行评估
        PerformEvolution ();               //进化
        OutputTextReport ();
    }
    return 0;
}

/*****function:generation the first popolation 初始化群体*****/
void GenerateInitialPopulation (void)
{
    int i,j;
    srand((unsigned)(time(NULL)));
    for (i=0;i<POPSIZE;i++)
    {
        for (j=0;j<CHROMLENGTH;j++)
        {
            population [i].chrom[j]=(random(10)<5)?'0':'1';
        }
        population [i].chrom[CHROMLENGTH]='\0';
    }
}

/*****function: generate the next generation 产生新种群*****/
void GenerateNextPopulation (void)
{
    SelectionOperator ();
    CrossoverOperator ();
    MutationOperator ();
}

/****function: evaluate population according to certain formula 衡量群体*****/
void EvaluatePopulation (void)
{
    CalculateObjectValue ();
    CalculateFitnessValue ();
    FindBestAndWorstIndividual ();
}
```

```
/*****function: to decode a binary chromosome into a decimal integer*****/
long DecodeChromosome (char *string,int point,int length)
{
	int i;
	long decimal=0L;
	char *pointer;
	for (i=0,pointer=string+point;i<length;i++,pointer++)
	{
		decimal+=(*pointer-'0')<<(length-1-i);
	}
	return (decimal);
}

/***** function:to calculate object value f(x) = x sin(10 π x) + 2.0 *****/
void CalculateObjectValue (void)
{
	int i;
	long temp;
	double x;
	/*** rosenbrock function***/
	for (i=0;i<POPSIZE;i++)
	{
		temp=DecodeChromosome (population[i].chrom,0,CHROMLENGTH);
		x=(RIGHT-LEFT)*temp/(pow(2,CHROMLENGTH)-1.0)+LEFT;
		population[i].value=x*sin(10*PI+x)+1.0;
		population[i].x=x;
	}
}

/******function: to calculate fitness value *******/
void CalculateFitnessValue (void)
{
	int i;
	for(i=0;i<POPSIZE;i++)
	{
		population[i].fitness=population[i].value;
	}
}

/*****function to find out the best individual so far current generation*****/
void FindBestAndWorstIndividual (void)
{
	int i;
```

```
    double sum=0.0;
    /*** find out the best and worst individual of this generation***/
    bestindividual=population[0];
    worstindividual=population[0];
    for(i=1;i<POPSIZE;i++)
    {
        if (population[i].fitness>bestindividual.fitness)
        {bestindividual=population[i];best_index=i;}
        else if(population[i].fitness<worstindividual.fitness)
        {worstindividual=population[i];worst_index=i;}
        sum+=population[i].fitness;
    }
    /***find out the best individual so far***/
    if (generation==0){ currentbest=bestindividual;}
    else
    {
        if(bestindividual.fitness>currentbest.fitness) {currentbest=bestindividual;}
    }
}

/*****function:to perform evolution operation based on elitise mode. elitist model is to
        replace the worst individual of this generation by the current best one 保留最优个体
*****/
void PerformEvolution (void)
{
    if(bestindividual.fitness>currentbest.fitness){currentbest=population[best_index];}
    else{population[worst_index]=currentbest;}
}

/*****function: to reproduce a chromosome by roulette wheel seclection*****/
void SelectionOperator (void)
{
    int i,j,index;
    double p,sum=0.0;
    double cfitness[POPSIZE];    /*cumulative fitness value*/
    struct individual newpopulation[POPSIZE];
    /***calculate relative fitness***/
    for(i=0;i<POPSIZE;i++) {sum+=population[i].fitness;}
    for(i=0;i<POPSIZE;i++){cfitness[i]=population[i].fitness/sum;}
    /***calculate cumulative fitness***/
    for(i=1;i<POPSIZE;i++){cfitness[i]=cfitness[i-1]+cfitness[i];}
    /***selection operation***/
    for(i=0;i<POPSIZE;i++)
    {
```

```
            p=random(1000)/1000.0;
            index=0;
            while(p>cfitness[index]){index++;}
            newpopulation[i]=population[index];
        }
        for(i=0;i<POPSIZE;i++){population[i]=newpopulation[i];}
}

/*****function:crossover two chromosome by means of one-point crossover*****/
void CrossoverOperator (void)
{
        int i,j;
        int index[POPSIZE];
        int point,temp;
        double p;
        char ch;
        /***make a pair of individual randomly***/
        for(i=0;i<POPSIZE;i++){index[i]=i;}
        for(i=0;i<POPSIZE;i++)
        {
            point=random(POPSIZE-i);
            temp=index[i];
            index[i]=index[point+i];
            index[point+i]=temp;
        }
        /***one-point crossover operation***/
        for(i=0;i<POPSIZE-1;i+=2)
        {
            p=random(1000)/1000.0;
            if(p<Pc)
            {
                point=random(CHROMLENGTH-1)+1;
                for(j=point;j<CHROMLENGTH;j++)
                {
                    ch=population[index[i]].chrom[j];
                    population[index[i]].chrom[j]=population[index[i+1]].chrom[j];
                    population[index[i+1]].chrom[j]=ch;
                }
            }
        }
}

/*****function: mutation of a chromosome*****/
void MutationOperator (void)
```

```
{
	int i,j;
	double p;
	/*** bit mutation***/
	for(i=0;i<POPSIZE;i++)
	{
		for(j=0;j<CHROMLENGTH;j++)
		{
			p=random(1000)/1000.0;
			if(p<Pm){population[i].chrom[j]=(population[i].chrom[j]=='0')?'1':'0';}
		}
	}
}

/*****function: output the results of current population*****/
void OutputTextReport (void)
{
	int i;
	double sum;
	double average;	/***average of population object value***/
	/*** calculate average object value***/
	sum=0.0;
	for(i=0;i<POPSIZE;i++){sum+=population[i].value;}
	average=sum/POPSIZE;
	/***print results of this population ***/
	printf("gen=%d, avg=%f, x=%f, best=%f
",generation,average,currentbest.x,currentbest.value);
	printf("chromosome=");
	for(i=0;i<CHROMLENGTH;i++){printf("%c",currentbest.chrom[i]);}
	printf("\n");
}
```

六 实验结果以及分析

```
Cmder

gaolex@HASEE:/mnt/c/Users/gaolex/Desktop/tyut-ai-homework$ g++ ga.cpp
gaolex@HASEE:/mnt/c/Users/gaolex/Desktop/tyut-ai-homework$ ./a.out
gen=1, avg=2.052903, x=1.912640, best=2.801988 chromosome=11111000100010111100101
gen=2, avg=2.262124, x=1.912817, best=2.802041 chromosome=11111000100011111011100
gen=3, avg=2.304695, x=1.997873, best=2.818447 chromosome=11111111110100011000001
gen=4, avg=2.506477, x=1.997873, best=2.818447 chromosome=11111111110100011000001
gen=5, avg=2.534801, x=1.997873, best=2.818447 chromosome=11111111110100011000001
gen=6, avg=2.575312, x=1.999397, best=2.818570 chromosome=11111111111100101101000
gen=7, avg=2.566340, x=1.999397, best=2.818570 chromosome=11111111111100101101000
gen=8, avg=2.612774, x=1.999397, best=2.818570 chromosome=11111111111100101101000
gen=9, avg=2.636198, x=1.999397, best=2.818570 chromosome=11111111111100101101000
gen=10, avg=2.633980, x=1.999397, best=2.818570 chromosome=11111111111100101101000
gen=11, avg=2.629552, x=1.999398, best=2.818570 chromosome=11111111111100101101010
gen=12, avg=2.640538, x=1.999426, best=2.818572 chromosome=11111111111100110111000
gen=13, avg=2.619738, x=1.999677, best=2.818592 chromosome=11111111111110001111000
gen=14, avg=2.647462, x=1.999979, best=2.818615 chromosome=11111111111111111000001
gen=15, avg=2.660610, x=1.999979, best=2.818615 chromosome=11111111111111111000001
gen=16, avg=2.679643, x=1.999997, best=2.818617 chromosome=11111111111111111111011
gen=17, avg=2.702213, x=1.999997, best=2.818617 chromosome=11111111111111111111011
gen=18, avg=2.711464, x=1.999997, best=2.818617 chromosome=11111111111111111111011
gen=19, avg=2.728719, x=1.999997, best=2.818617 chromosome=11111111111111111111011
gen=20, avg=2.720216, x=1.999997, best=2.818617 chromosome=11111111111111111111011
gen=21, avg=2.719240, x=1.999997, best=2.818617 chromosome=11111111111111111111011
gen=22, avg=2.724087, x=1.999999, best=2.818617 chromosome=11111111111111111111101
gen=23, avg=2.737641, x=1.999999, best=2.818617 chromosome=11111111111111111111101
gen=24, avg=2.741111, x=1.999999, best=2.818617 chromosome=11111111111111111111101
gen=25, avg=2.732414, x=1.999999, best=2.818617 chromosome=11111111111111111111101
gen=26, avg=2.741624, x=1.999999, best=2.818617 chromosome=11111111111111111111101

bash.exe
```

由此可见，f(x)=x*sin(10π+x)+1.0 在区间[-1，2]的最大值约为 2.817.这与实际函数图像（见下图）所示相符合。