北京航空航天大學
BEIHANG UNIVERSITY

模式识别与智能系统技术

人工智能部分大作业

院(系)名称	高等理工学院
学　　号	16231235
姓　　名	李谨杰
指导教师	王　岩

2019 年 5 月

一、实验目的

1、使学生加深对图搜索技术的理解
2、掌握图搜索基本编程方法
3、运用图搜索技术解决一些应用问题

二、实验要求

1、用启发式搜索算法实现野人过河问题。
2、有明确的状态空间表达，规则集以及估计函数。
3、程序运行时，应能清晰直观演示搜索过程。

三、实验内容

（一）原理分析

启发式搜索算法分析：

步 1 把初始节点 So 放入 OPEN 表中, 计算 f(So)。

步 2 若 OPEN 表为空,则搜索失败, 退出。

步 3 移出 OPEN 表中第一个节点 N 放入 CLOSED 表中, 并冠以序号 n。

步 4 若目标节点 Sg=N, 则搜索成功, 结束。

步 5 若 N 不可扩展, 则转步 2。

步 6 扩展 N, 计算每个子节点 x 的函数值 f(x), 并将所有子节点配以指向 N 的返回指针后放入 OPEN 表中, 再对 OPEN 表中的所有子节点按其函数值大小以升序排序,转步 2。

启发式在搜索中，若节点的估价函数定义方式满足：f(n)= g(n)+)+h(n)，h(n)≤h*(n)，则该算法是 A*算法，是可采纳的。

（二）实验内容

传教士和野人问题。有三个传教士和三个野人一起来到河边准备渡河，河边有一条空船，且传教士和野人都会划船，但每次最多可供两人乘渡。河的任何一岸以及船上一旦出现野人人数超过传教士人数，野人就会把传教士吃掉。为安全地渡河，传教士应该如何规划渡河方案？试给出该问题的状态图表示，并编程求解之。

若传教士和野人的数码均为 5 人，渡船至多课程 3 人，请定义一个启发函数，并给出相应的搜索树。

四、实验步骤

具体工作及步骤为：

1. 设计问题的状态表示方法；
2. 根据相应的状态表示方法，给出左岸到右岸的渡河移动规则；
3. 实现状态空间图；
4. 定义该问题的启发式函数，判断该定义是否满足 A*算法？
5. 实现搜索过程，分析实验结果，给出搜索图。
6. 讨论 5 人问题时，搜索图的变化以及搜索问题的复杂程度，给出搜索树。
7. 撰写实验报告，对启发式搜索算法给出个人的总结分析，写出实验感受。

五、实验结果

1. 设计状态表示方法如下：

第一列为左岸的状态，第二列为右岸的状态；第一行为传教士的人数，第二行为野人的人数，第三行为船停泊的状态。1 表示有船停泊，0 表示没有船停泊。对于三人问题，初始状态和目标状态如下：

$$S_0=\begin{pmatrix}3 & 0\\ 3 & 0\\ 1 & 0\end{pmatrix} \qquad S_g=\begin{pmatrix}0 & 3\\ 0 & 3\\ 0 & 1\end{pmatrix}$$

2. 给出左岸到右岸的渡河规则：

定义船的状态为 $B=\begin{pmatrix}C\\ Y\\ S\end{pmatrix}$

第一行表示传教士数量，第二行表示野人数量，第三行用于改变船的状态，保持常数 1 不变。船的状态共有$[1,0,1]^T$, $[0,1,1]^T$, $[1,1,1]^T$, $[2,0,1]^T$, $[0,2,1]^T$ 五个状态，从左岸开往右岸后，进行如下判断：

（1）判断左岸已有的人数可否支持船的状态 B，不满足则舍弃该船状态 B。

（2）如果支持，原状态矩阵 S 第一列减去 B 向量，状态矩阵 S 第二列加上 B 向量，得到改变后的状态矩阵 S1。

（3）判断新状态的两列是否满足传教士人数大于等于野人数的条件，如果不满足就舍去该状态。

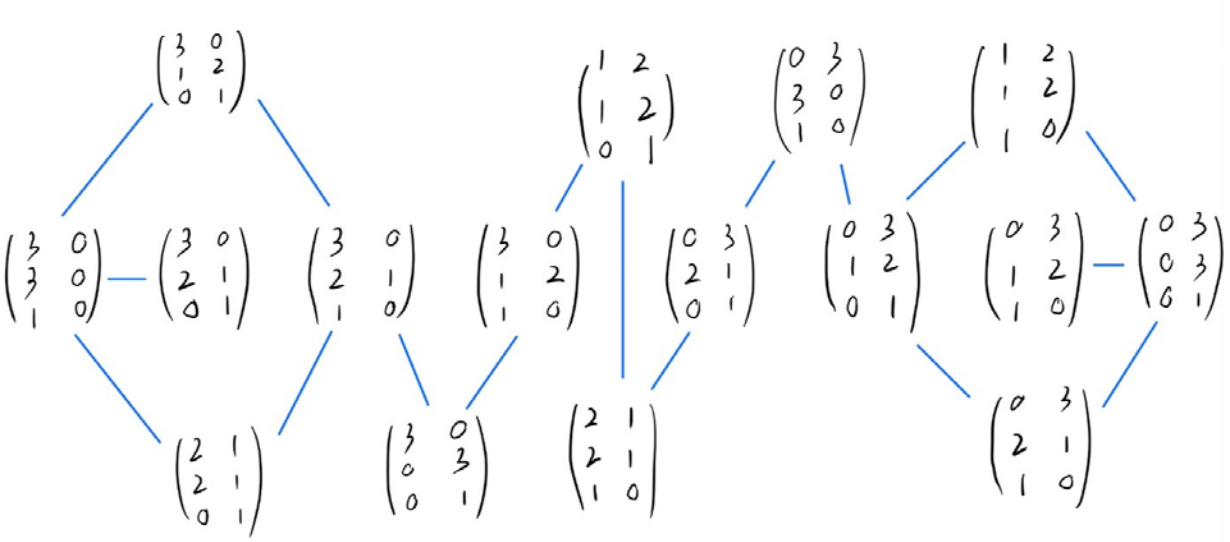
（4）判断新状态与先辈节点是否相同，相同则舍去该状态。

（5）改变 B 状态回到（1）；如果遍历完全部船状态，退出循环。

右岸到左岸的规则与此类似。在 MATLAB 程序中，渡河规则使用 plusyeah.m 和 plusjudge.m 函数完成。

3. 实现状态空间图

最左边为 S0，最右边为目标 Sg



4. 定义该问题的启发式函数

定义启发函数 1：

h=目前状态矩阵与目标状态矩阵之间不同数字的个数。

例如：

$$S_i=\begin{pmatrix}0 & 3\\ 1 & 2\\ 0 & 1\end{pmatrix} \qquad S_g=\begin{pmatrix}0 & 3\\ 0 & 3\\ 0 & 1\end{pmatrix}$$

不一样的数字为第二行的 1,2，故 h(i)=2。通过程序可以求出第 12 个节点的启发值 h(12)=4，但只需要 1 步就能解决，故不满足 A*算法。

定义启发函数 2：

h=目前状态矩阵与目标矩阵对应元素的距离之和。

$$S_i=\begin{pmatrix}3 & 0\\ 1 & 2\\ 1 & 0\end{pmatrix} \qquad S_g=\begin{pmatrix}0 & 3\\ 0 & 3\\ 0 & 1\end{pmatrix}$$

对上面的例子，从左往右，从上到下不同元素的距离分别是 3,3,1,1,1,1，故 h(i)=3+3+1+1+1+1=10。 通过该定义可以求出第 6 个节点 h(6)=10，但距目标深度为 7，故不满足 A*算法。

定义启发函数 3：

$h = [\frac{2M+2N-(K+1)B}{K-1}]$，[]表示向上取整。其中 M 为左岸传教士的人数，N 为左岸野人的人数，K 为船最多载人数，B 为船的状态，如果船在左岸 B=1，否则为 0。

$$S_i=\begin{pmatrix}3 & 0\\ 1 & 2\\ 1 & 0\end{pmatrix} \qquad S_g=\begin{pmatrix}0 & 3\\ 0 & 3\\ 0 & 1\end{pmatrix}$$

对状态 Si，h(i)=6+2-3=5。从搜索树结果来看，h 值均小于实际路径，故该算法是 A*算法。

这三个启发函数里，1 和 2 是我自己定义的，3 是根据网上查到的初步算法（见参考文献 3）推导而来的，是适合所有人数的普适性算法。我还尝试了一些其他自己设定的算法，但自己设定的算法均无法满足 A*要求。可见 A*算法需要以理论依据为基础，推导过程见部分 6。

5. 实现搜索过程，分析实验结果，给出搜索图。

我实现了深度优先搜索，广度优先搜索，A 算法，A*算法。选用启发式函数为：h=目前的状态与目标状态不一样的数字的个数（A 算法），将启发式搜索的状态空间图绘制如图 5.1：（MATLAB 中序号从 1 开始，故程序中的状态标号比这里的加 1）。其他算法的搜索过程与该图类似，不同的地方是 f 值、h 值和每个节点的标号与状态。为看得更清楚，我绘制了第一张图，其余搜索图由 matlab 生成。

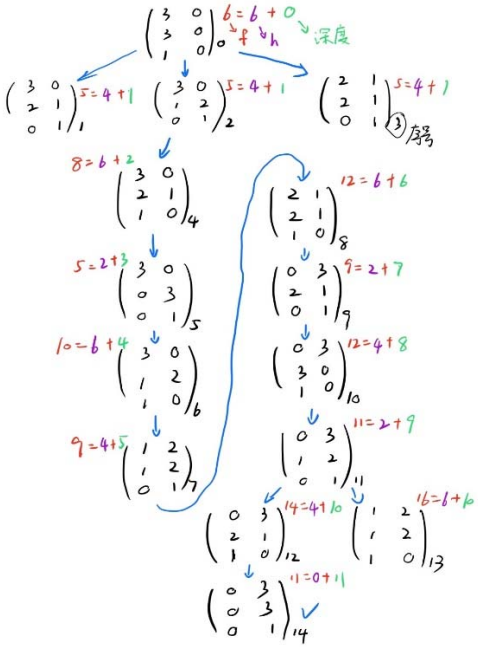


图 5.1 三人问题启发式算法搜索图

A*算法：

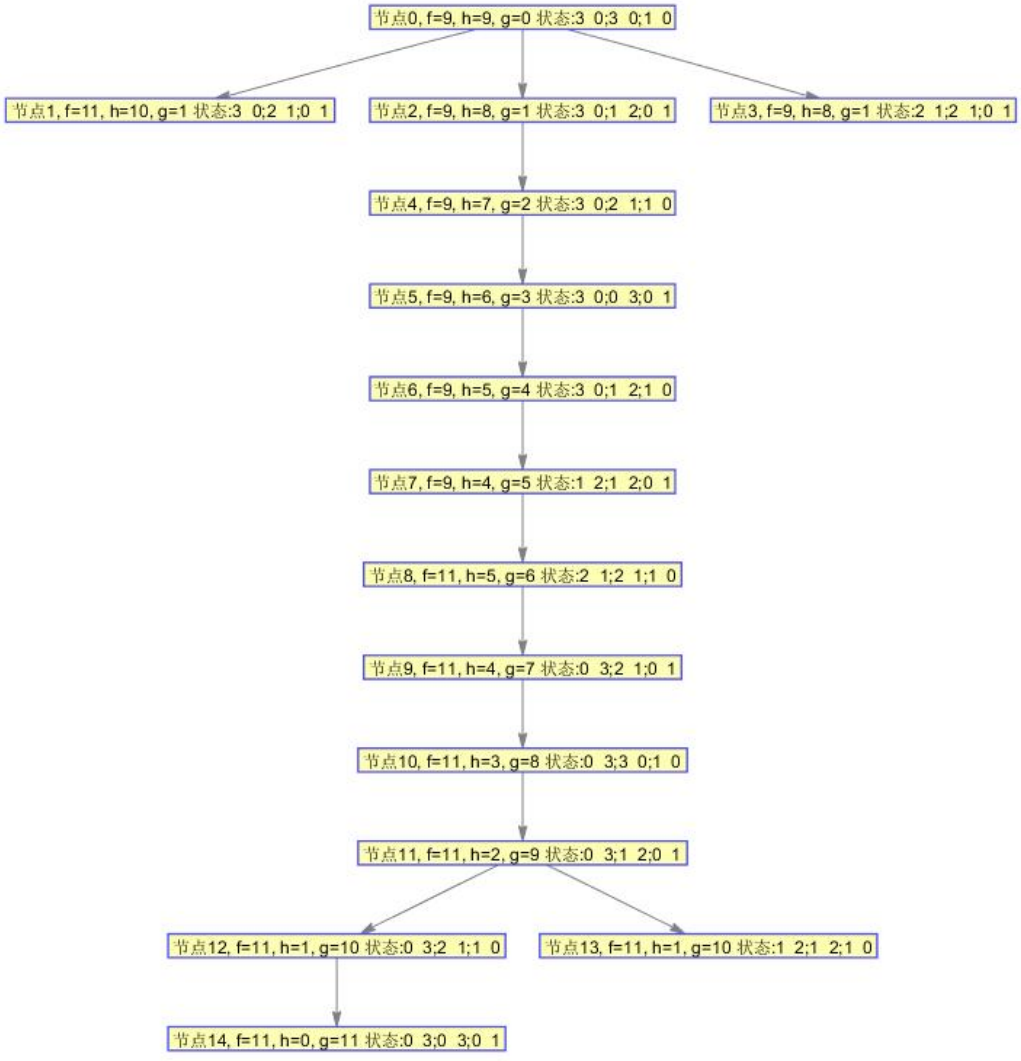


图 5.2 三人问题 A*算法搜索图

由上图 h 的变化可知该启发式算法满足 A*算法的要求。

三人问题几种算法对比如下：

算法名称	深度	计算节点数
广度优先	11	15
深度优先	11	12
A 算法 1	11	14
A 算法 2	11	14
A*算法	11	14

可见对于解较少的问题，不同算法的差异并不大。

6. 五人问题时，不同算法对比如下：

算法名称	深度	计算节点数
广度优先	11	26
深度优先	13	16
A 算法 1	11	23
A 算法 2	11	19
A*算法	11	20

注：由于用启发式算法从 open 表里取点时，f 值相同的几个点取第一个，这样新生成的节点放进 open 表的方式，会导致计算节点数产生微小差异。本表格是按照放在 open 表尾部计算的。

变为五人问题后，状态比三人问题多了一些，但最优路径的深度没有改变。在五人问题中，各种算法的特点展现地更加彻底。搜索图如下：

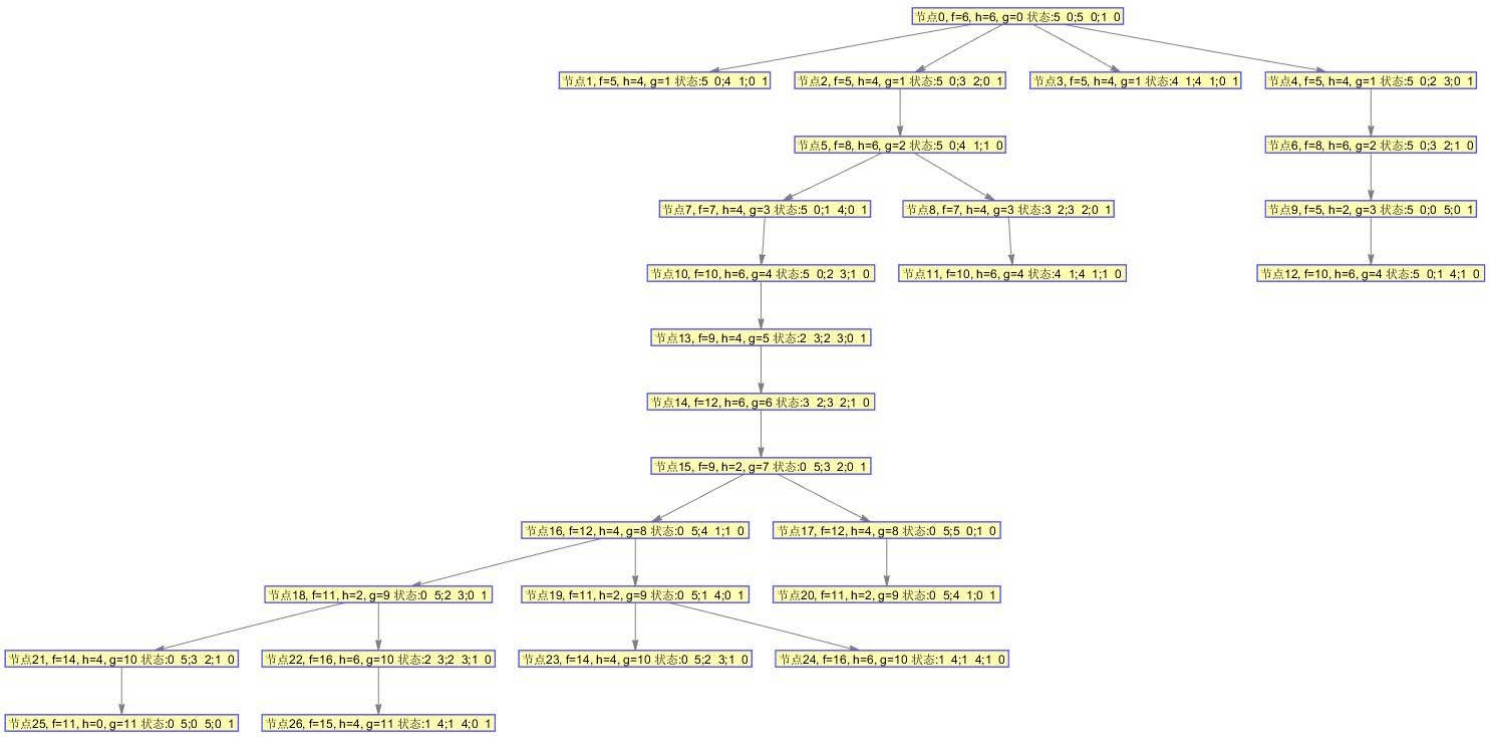


图 6.1 广度优先搜索图

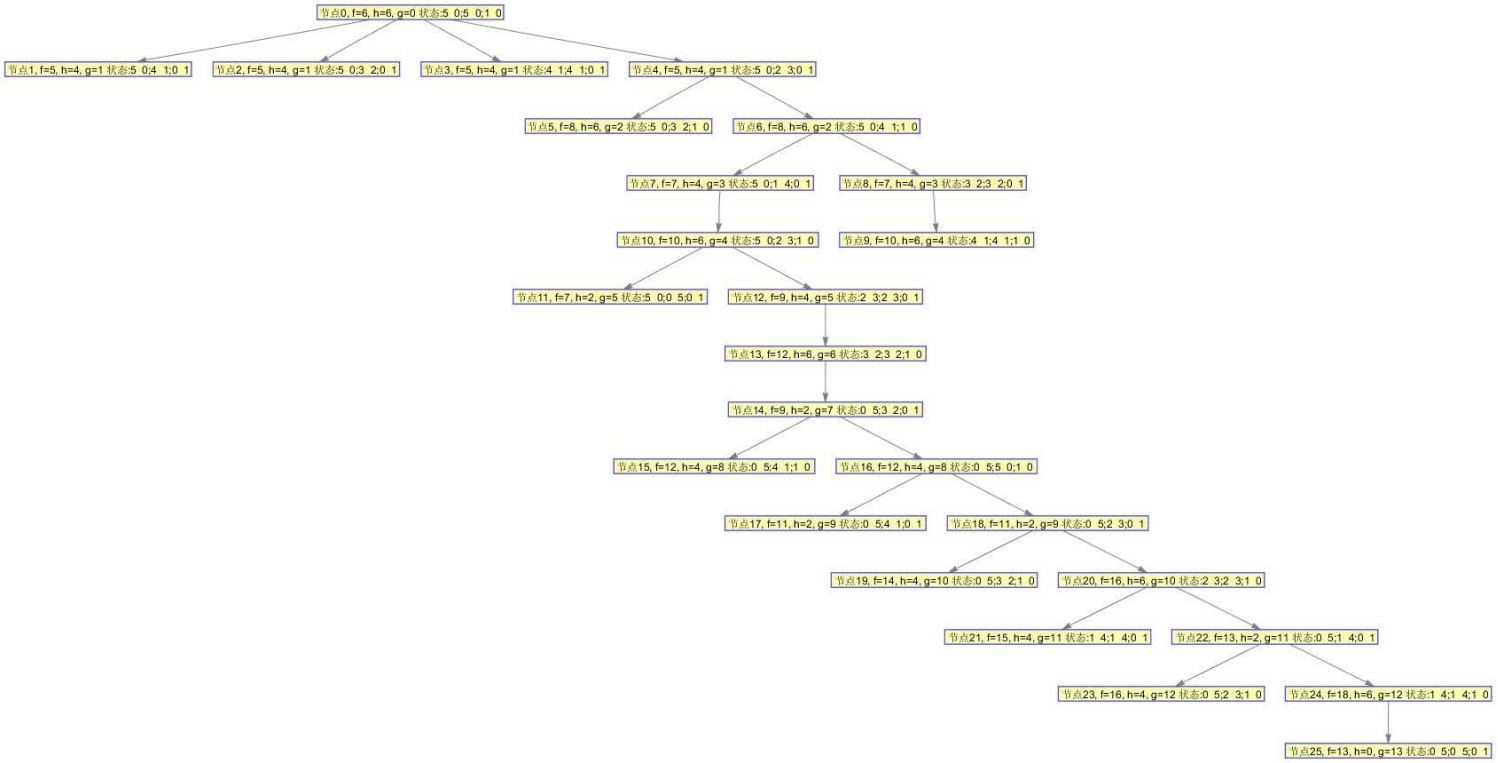


图 6.2 深度优先搜索图

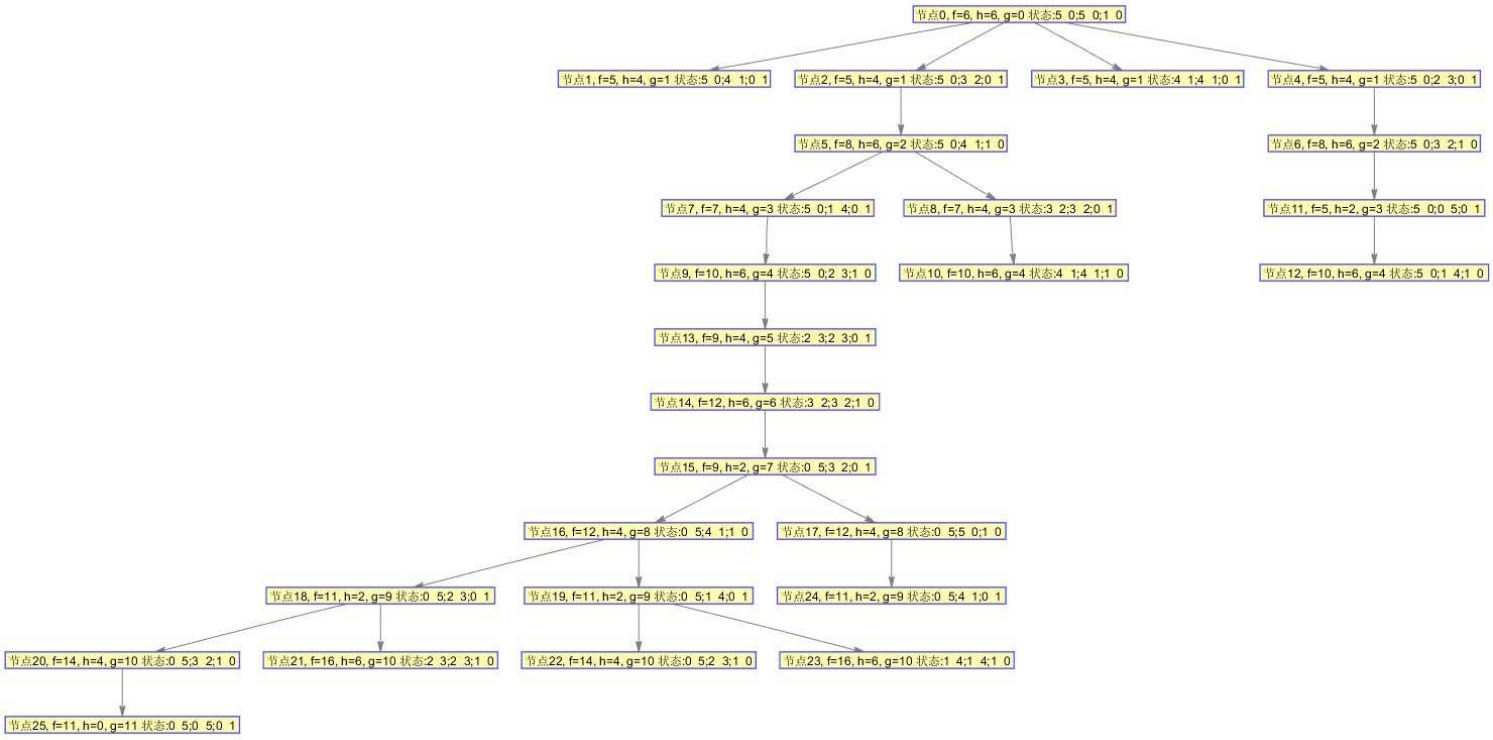


图 6.3 第一种 A 算法搜索图

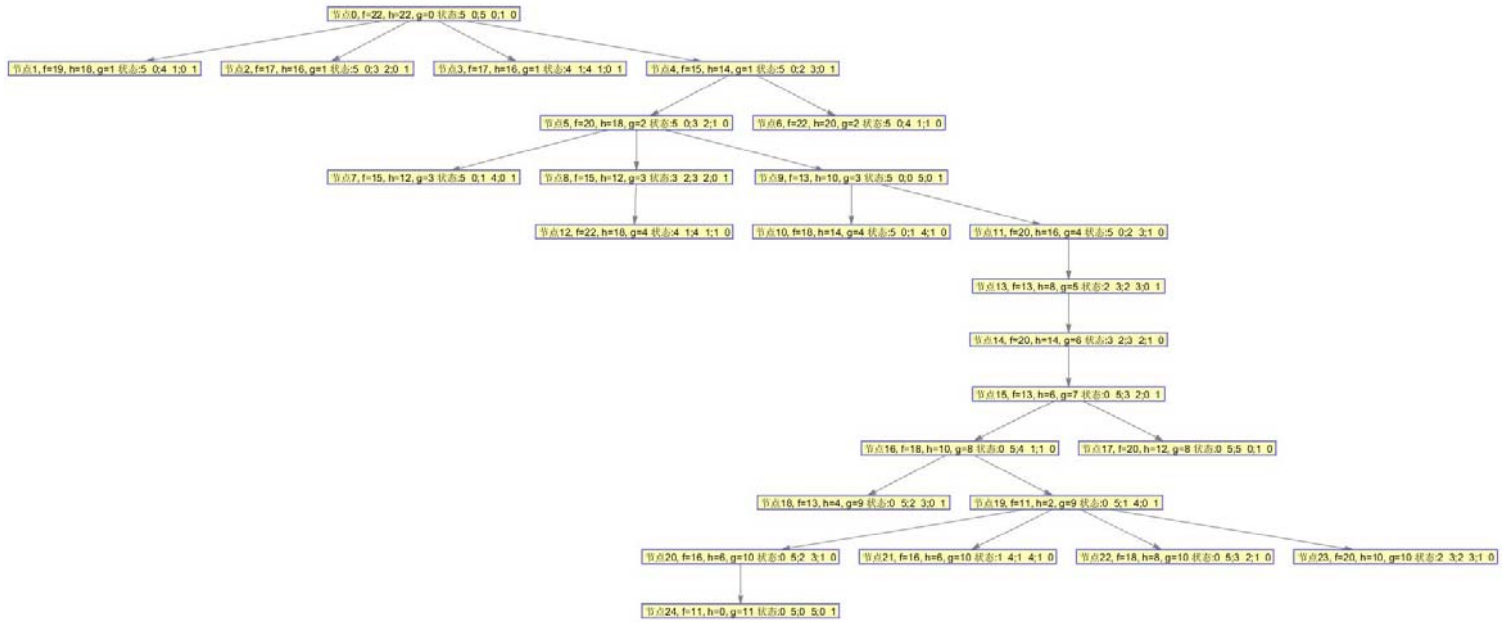


图 6.4 第二种 A 算法搜索图

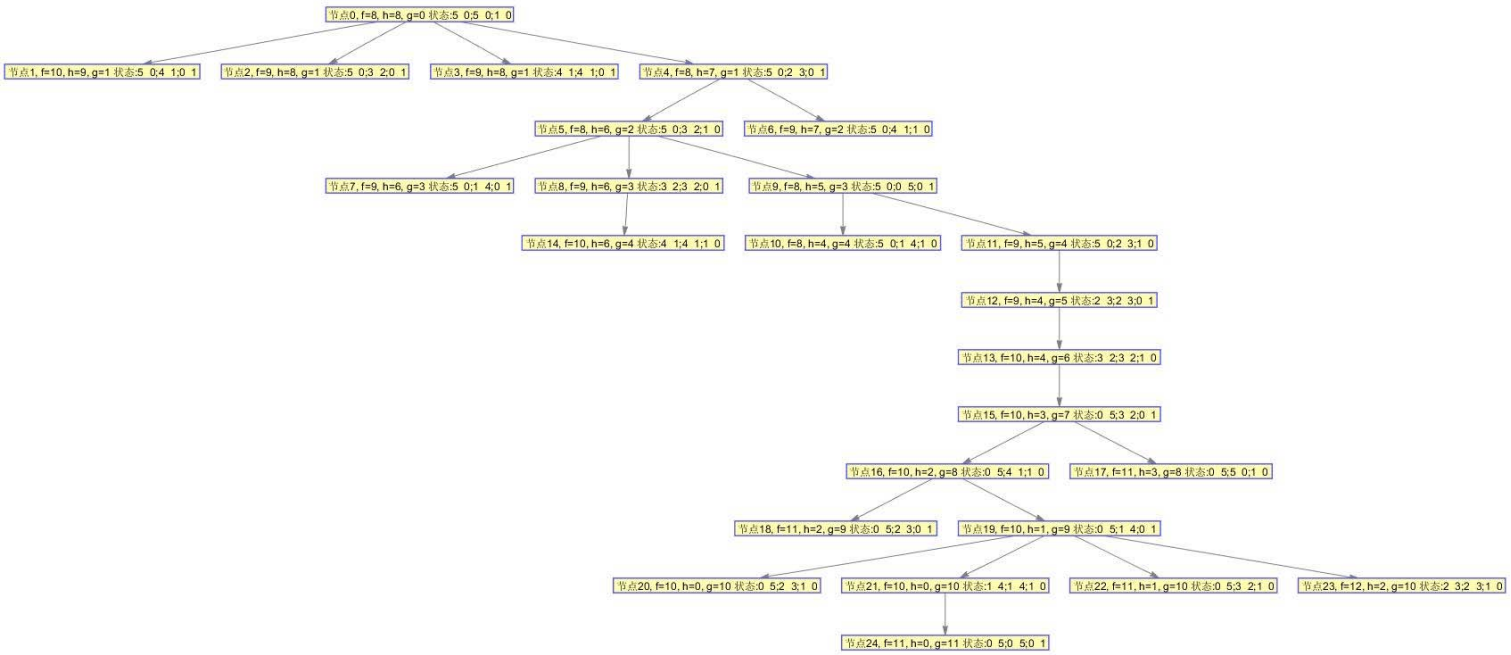


图 6.5 A*算法搜索图

综合表格和搜索图，可以得到以下结论：

（1） 如果不存在很深的无解的分支，深度优先计算节点数最少，速度最快。

（2） 广度优先计算节点数最多，速度是最慢的，但可以达到最优解。

（3） A*、A 算法的速度介于深度优先和广度优先算法之间，但 A 算法不能保证一定达到最优解。

（4） 深度优先算法得到的深度是最深的。

A*算法推导：

假设 M 为左岸传教士人数，N 为左岸野人人数，B 为左岸船的状态,船最多

可以运送 K 个人。考虑船在左岸，如果不考虑限制条件，船一个来回可以运送（K-1）个人，而船仍在左岸。最后剩下 K 个人，可以一次运过去。所以，即使不考虑限制条件也需要摆渡$2*\frac{M+N-K}{K-1}+1=\frac{2M+2N-(K+1)}{K-1}$次。再考虑船在右岸，此时需要一个人把船运回去。此时相当于摆渡 1 次+左岸为（M+1，N）状态时需要的摆渡数，即$\frac{2(M+1)+2N-K-1}{K-1}+1=\frac{2(M+N)}{K-1}$次。因为 h 必须是整数，故综合两个情况，可以定义$h=[\frac{2M+2N-(K+1)B}{K-1}]$，在左岸 B 为 1，右岸 B 为 0,向上取整。由于加入限制条件后，摆渡次数肯定多于 h，因此该算法从理论上是 A*的。另外，除题设要求外，我还试验了 6 传教士、6 野人、船载 4 人和 7 传教士、7 野人、船载 4 人的情况，均满足 A*条件。

A*算法一定要结合实际情况进行数学推导，切忌自己试验。

六、收获、体会及建议

因为网上的野人过河代码均为 C++语言编写，我并不会，故本次作业我是只看着 PPT 的算法写的，中间经历了无数 debug 的过程，最后感觉还是很有成就感的！一开始我先实现了广度优先算法和深度优先算法，之后更改 open 表取出扩展条件和 h 函数，得到了 A 算法和 A*算法的结果。之后我又修改了 A*算法，优化程序，使得对于任意人数情况均可以计算。最后我查了查 matlab 绘制树状图的方法，使得可以同步生成搜索图。程序运行过程中可以一直显示扩展生成的节点状态，用以表征动态搜索过程。

经过本次实验，我完全理解了老师讲的搜索算法。我对 A*算法选取的经验是，一定要基于理论推导，不能随意指定。报告最后附实验代码。

如果对报告或代码有任何问题，麻烦老师联系我，E-mail：lijinjie362@outlook.com，手机 or 微信：15652587808，谢谢老师！

参考文献

[1] 人工智能技术导论（第三版）廉师友 西安电子科技大学出版社
[2] 人工智能课程讲义：第一章图搜索 王岩
[3] 修道士和野人过河问题 A*算法 人工智能
https://blog.csdn.net/yaling521/article/details/38825389
[4] Matlab 绘制树形图 https://blog.csdn.net/C_Redrock/article/details/84980241

Main.m

```
% 功能：计算野人过河问题并给出每一步的状态
clear;clc;

isqfs=1;  %1 启发式；0 穷举式
method=3;  %1 不在位个数；2 相差绝对值 3；  A*  4. 不包含船的相差绝对值
isdeep=0; %1 深度优先；  0 广度优先
如果选了启发式这个选项没用

num_cjs=7;  %传道士人数
num_yr=7;   %野人人数
num_boat=4; %船上人数
set=struct('cjs',num_cjs,'yr',num_yr,'boat',num_boat','method',method);

num_count=0;  %统计计算了几个节点
S(:,:,1)=[num_cjs,0;num_yr,0;1,0];  %man;beast;boatstate  left and right 初始状态为S1
Sg=[0,num_cjs;0,num_yr;0,1];
op=[];cl=[];  %初始化 open ; closed
Lo=0; %length of open
Lc=0; %length of closed
id=1; %symbol 每一个节点的标号
success=0; % if success, 1
Dp=0; %表示当前搜索的深度
fuDp=-1; %仅供初始化第一个矩阵
depth(1)=Dp;
M(1) =
IniM(0,S(:,:,1),fuDp,Sg,set);  %初始化过程中已经计算f
%估价函数f(i)=h(i)+depth
%%
%把初始节点S0放入Open表，计算f（S1）
op(1)=1;Lo=Lo+1;
%%
while(Lo~=0)   %若open表为空，退出
   num_count=num_count+1; %计算次数加一
   %%
   if isqfs == 1
      %启发式算法启用
      fmin=M(op(1)).f;  %fmin是open表第一个值
      mintag=1;
      for i=1:Lo   %找出估价函数最小的mintag,多个f值相同选第一个tag
         if M(op(i)).f<fmin
            fmin=M(op(i)).f;
            mintag=i;
         end
      end


cl=[op(mintag),cl(1:end)];Lc=Lc+1; %更新closed

      if mintag == 1          %更新open
         op=op(mintag+1:end);
      else if mintag == Lo
         op=op(1:mintag-1);
         else
         op=[op(1:mintag-1),op(mintag+1:end)];
         end
      end
      Lo=Lo-1; %将op表中第一个节点放在closed表中
   %%
   else
%  深度优先，广度优先时启用

cl=[op(1),cl(1:end)];Lc=Lc+1; %更新closed
       op=op(2:end);Lo=Lo-1; %将op表中第一个节点放在closed表中
   end
   %%
   if isequal(M(cl(1)).State,Sg)
      success=1;
      break;   %搜索成功，退出
   end
```

```
[N,j]=plusyeah(M,cl(1),num_boat); %N为需要判断的新生成的子状态，j为生成的子状态数
   if j == 0 %不可拓展 下一轮
      continue
   end
   fuDp=M(cl(1)).g;  %父节点深度
   fuPtr=cl(1);      %父节点的标号
  %%
   for q=1:j  %遍历所有新状态
      iscopy=0;  %表示是否与open表或closed表重复
      if Lo~=0   %如果open表不为空
         for temp=1:Lo  %检查所有open表
            if isequal(M(op(temp)).State,N(:,:,q)) %如果有子状态和open表中的一致
               iscopy=1; %只要有一致，iscopy=1
               if M(op(temp)).g>fuDp+1  %如果新路径的深度较短，这里改为启发式算法也不用管，因为相同状态，f值的差异在于深度

M(op(temp))=IniM(fuPtr,M(op(temp)).State,fuDp,Sg,set); %修改返回指针，修改深度
                  op = Renew_op(op(temp),op,isdeep);
Lo=Lo+1; %放回open表某处，各种算法不一样的地方
               end
               break; %只要找到一致的，就跳出open表寻找循环
            end
         end
          if iscopy
             continue;
          end
      end
      %%
      for temp=1:Lc  %检查所有closed表
         if isequal(M(cl(temp)).State,N(:,:,q)) %如果有子状态和open表中的一致
            iscopy=1;
            if M(cl(temp)).g>fuDp+1  %如果新路径的深度较短

M(cl(temp))=IniM(fuPtr,M(cl(temp)).State,fuDp,Sg,set); %修改返回指针，修改深度

op=Renew_op(cl(temp),op,isdeep);
Lo=Lo+1; %放入open表中重新拓展
            end
            break; %只要找到一致的，就跳出open表寻找循环
         end
      end
      if iscopy
         continue;
      end
      %%
      id=id+1;  %增加一个节点

M(id)=IniM(fuPtr,N(:,:,q),fuDp,Sg,set);

op=Renew_op(id,op,isdeep);Lo=Lo+1; %放回open表首部重新扩展

   end
end
%%
%显示

for i=1:id
   disp(['标号: ',num2str(i)]);
   disp(['父节点标号: ',num2str(M(i).Ptr)]);
   disp(['估价f=',num2str(M(i).f)]);disp(['启发
```

```
h=',num2str(M(i).h)]);disp(['深度
g=',num2str(M(i).g)]);

disp(M(i).State);%fprintf('%c%c'
, 8, 8);
end
disp(['计算节点数:
',num2str(num_count)]);
if success
    disp('Success!');
else
    disp('No solution!');
end

%treeplot(tree);作图

tree=zeros(id-1,2);
juzhen=[' 状
态:',num2str(M(1).State(1,:)),';'
,num2str(M(1).State(2,:)),';',nu
m2str(M(1).State(3,:))];
ChannelName(1)={['节点
',num2str(0),',
f=',num2str(M(1).f),',
h=',num2str(M(1).h),',
g=',num2str(M(1).g),juzhen]};
for i=2:id
    tree(i-1,1)=M(i).Ptr-1;
    tree(i-1,2)=i-1;
    juzhen=[' 状
态:',num2str(M(i).State(1,:)),';'
,num2str(M(i).State(2,:)),';',nu
m2str(M(i).State(3,:))];
    ChannelName(i)={['节点
',num2str(i-1),',
f=',num2str(M(i).f),',
h=',num2str(M(i).h),',
g=',num2str(M(i).g),juzhen]};
end
cm = zeros(id);
for i = 1:id-1
   cm(tree(i,1)+1,tree(i,2)+1) =
1;
end
bg1 = biograph(cm,ChannelName);
view(bg1);
```

get_h.m

```
function h = get_h(S,Sg,set)
%函数功能：计算估价函数
%输入：现在状态S，目标状态Sg
%输出：估价值h
    method=set.method;
    if method == 1
    %估价方法1，不一样的个数
    A=S-Sg;
    A(A~=0)=1;
    h=sum(sum(A));
    else if method ==2
            %估价方法2，不一样的数字相
减，求和绝对值
            A=S-Sg;
            A=abs(A);
            h=sum(sum(A));
        else if method ==3
            %估价方法3，推出来的

h=ceil((2*(S(1,1)+S(2,1))-
(set.boat+1)*S(3,1))/(set.boat-
1));
            else
                A=S(1:2,:)-
Sg(1:2,:);
                A=abs(A);
                h=sum(sum(A));
            end
        end
    end
end
```

IniM.m

```
function M =
IniM(fuPtr,A,fuDp,Sg,set)
%函数功能：为M初始化
%输入：父标号，状态值，父节点的深度，h
值
%输出：初始化的结构体M
h=get_h(A,Sg,set);
Dp=fuDp+1; %现有深度比父节点加一
```

```
M=struct('Ptr',fuPtr,'State',A,'f',h+Dp,'h',h,'g',Dp);
End
```

plusjudge.m

```
function [Nt,j] = plusjudge(K,a,B)
%函数功能：判断是否是先辈节点，判断是否野人会吃掉传教士
%输入：结构体序列K，要扩展的状态标号a，船的状态B
%输出：子状态（若有，没有为0矩阵）Nt，子状态个数（1或0）

j=0;

if K(a).State(1,1)>=B(1) && K(a).State(2,1)>=B(2) && K(a).State(3,1)==1 %left to right
   Nt=K(a).State;
   Nt(:,1)=Nt(:,1)-B;Nt(:,2)=Nt(:,2)+B;
   if (Nt(1,1)>=Nt(2,1)||Nt(1,1)==0) && (Nt(1,2)>=Nt(2,2)||Nt(1,2)==0)  %左岸传教士大于等于野人，右岸同,或者传教士人数为0
      if K(a).Ptr==0   %没有先辈节点
          j=1;
      else if isequal(K(K(a).Ptr).State,Nt)~=1 %与先辈节点不同
             j=1;
         end
      end
   end

else if K(a).State(1,2)>=B(1) && K(a).State(2,2)>=B(2) && K(a).State(3,2)==1  %right to left
   Nt=K(a).State;

Nt(:,1)=Nt(:,1)+B;Nt(:,2)=Nt(:,2)-B;
   if (Nt(1,1)>=Nt(2,1)||Nt(1,1)==0) && (Nt(1,2)>=Nt(2,2)||Nt(1,2)==0)  %左岸传教士大于等于野人，右岸同,或者传教士人数为0
         if K(a).Ptr==0   %没有先辈节点
             j=1;
         else if isequal(K(K(a).Ptr).State,Nt)~=1 %与先辈节点不同
                j=1;
            end
         end
   end
    else
       Nt=zeros(3,2);
    end
end
end
```

plusyeah.m

```
function [N,j] = plusyeah(K,a,num_boat)
%函数功能：生成子状态，去除掉与父节点相同的状态和吃人状态
%输入：结构体序列，要扩展的状态标号
%输出：生成的子状态序列N，子状态序列个数j，如果没有子状态j=0

   j=0; %表示生成的子节点数目 N为获取的新状态
   %%
   B_total=0;
   for i=1:num_boat  %船上总人数i
      for k=0:i    %传道士的数量k
         if k>=(i-k) || k==0
```

```
            B_total=B_total+1;
            B(:,B_total)=[k;i-k;1];
            end
        end
    end


    if K(a).State(3,1) == 1
    %left to right
        for i=1:B_total
            [Nt,j1] =
plusjudge(K,a,B(:,i));
            if j1==1
                j=j+1;N(:,:,j)=Nt;
            end
        end
    end
    %%
    if K(a).State(3,1) == 0
            % right to left
        if j == 0    %表示不属于左岸,
计算右岸；若左岸已有，则不算右岸
            for i=1:B_total
            [Nt,j1] =
plusjudge(K,a,B(:,i));
                if j1==1

j=j+1;N(:,:,j)=Nt;
                end
            end
        end
    end
    if j==0   %没有生成的子节点，返回
空矩阵
            N=zeros(3,2);
    end
end

Renew_op.m

function opnew =
Renew_op(id,op,isdeep)
%函数功能：更新open表
%输入：添加的标签号，open表
%输出：更新后的open表
%注，这个改成启发式算法不用管，因为启发
式算法算最小值不看排序
if isdeep
    opnew=[id,op(1:end)]; %深度优
先,people=3,计算了12个节点
else
    opnew=[op(1:end),id]; %广度优
先,people=3,计算了15个节点
end
end
```