《华容道》游戏的搜索策略

李学武

（天津师范大学计算机系　天津 300074）

摘　要　本文利用人工智能产生式系统，采用双向广度优先搜索给出了求解《华容道》问题在一定意义下最优解的一个算法。本文首先介绍了该算法的基本思想，然后进行了复杂性分析，最后就源程序的核心部分给出适当的说明。本文作者还指导学生用 DELPHI 5.0 开发了相应的电脑游戏。部分相关资料已公诸于天津师范大学网站中由本文作者建立和主持的《信息学奥林匹克园地》栏目。

关键词　产生式系统　广度优先搜索　华容道　算法　复杂性分析　电脑游戏

A search strategy for playing Huarong Road

Li Xuewu

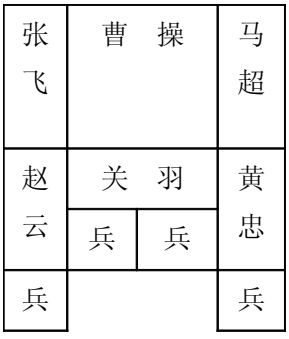
(Dept. of Computer Science, Tianjin Normal University, Tianjin , 300074)

Abstract: the paper presents an optimal algorithm for solving a game named “Huarong Road”. In this paper, basic steps on the algorithm was introduce first, and second it proceed to an analysis on algorithm complexity , finally, it remarked some kernel codes in the program. Otherwise, the author directed a student that developed a computer game under Borland Delphi 5.

Keywords: production system, breadth first search, Huarong Road, algorithm, complexity analysis, computer game

1　问题简介

华容道是在我国流传久远的一个益智游戏，然而其魅力至今不减，目前在许多商场内仍可见到基于这一游戏的玩具。该游戏起源于三国时期的一个著名故事：东汉末年，赤壁之战，曹操被周瑜杀得大败，带残兵从华容道仓皇逃走，不料大将关羽带兵在此等候。由于曹操与关羽曾经有过一段交往，关羽放曹操逃离华容道。华容道游戏的棋盘是由 20 个小正方形组成的长方形，宽 4 格，长 5 格。共有 10 个棋子，详见右下图。棋盘的下部有两个空置的小格作为华容道的出口。棋子就在这个长方形的棋盘内滑动，滑动过程中棋子不允许重叠。华容道是一个比较复杂的游戏，要移动很多步才能完成。据有关资料介绍[2]，对于上面的布局（华容道游戏还有多种布局），已知的最好走法是 81 步。



笔者利用典型的人工智能产生式系统[1]，采用双向广度优先搜索的方法搜索出在一定意义下最佳的解题步骤，在此基础上指导我系学生姚刚用 DELPHI 5.0 开发了一个相应的电脑游戏。本文首先介绍了该算法的基本思想，然后进行了复杂性分析，最后就 PASCAL 源程序的核心部分给出适当的说明与注释。限于

篇幅，许多重要资料（如完整的PASCAL源程序、搜索的结果、相应的电脑游戏等）不得不舍弃，部分相关资料可在天津师范大学网站（http://www.tjnu.edu.cn）的《信息学奥林匹克园地》栏目中找到，该栏目是由笔者建立和主持的。值得一提的是，我系学生姚刚开发的电脑游戏具有多种功能，除了让玩家自己动手使曹操脱离险境之外，还可完成诸如存储记录、读取记录、背景音效、选局、演示、求助计算机等功能。

2 搜索算法简介

我们用一个5*4的数组来记录棋盘状态，用标号1-5，8，9来表示各个棋子，例如，初始状态可表示为

4 9 9 5

4 9 9 5

2 8 8 3

2 1 1 3

1 0 0 1

（单向）广度优先搜索是大家熟知的算法，其基本步骤如下：

（2.1）将初始结点入队。队头指向该结点，队尾指向该结点之后。

（2.2） 将队头结点出队，生成所有可能的扩展结点，队头指向下一个结点。如果某个新生成的结点已符合目标则结束，否则通过比较，将与已在队列中的结点不重复的新结点入队。队尾指向新结点之后。

（2.3）重复（2.2），直到某个结点符合目标或队列空（即队尾位于队头之前）为止。

我们采用的双向广度优先搜索是指建立两个队列，分别从初始状态和目标状态出发进行搜索，直到遇到一个共同的结点（状态）为止。与单向搜索相比，其效率要高得多（详见后面的分析）。但这一方法必须要事先知道目标状态。实际计算表明，与单向搜索相比，我们的双向搜索方法可节省一半的空间，而时间不到前者的十分之一。因而是一个很有效的算法。

从理论上讲，广度优先搜索得到的结果在某种确定的意义下应该是最优的。

3 复杂性分析

在具体搜索过程中，两个空格（即数字0表示的方块）既可独立移动，又可同时移动，为简单计，在下面的讨论中，我们只考虑一个空格移动的情形。对于要扩展的节点，空格的不同方向的移动可产生新的节点，去掉一个父节点，还可产生3个新的节点（这里，暂不考虑可能与队列中已有点重复的点），假定按广度优先搜索，初始状态经过K步可达到目标状态，共需搜索K层，记初始状态为第0层，则第i层有节点3^i个，从第0层到第K层共有节点 $1+3+3^2+\cdots+3^K=(3^{K+1}-1)/2$ 个。对于单向广度优先搜索，除第K层可减少一些节点外，基本上是全部搜索，记

$$SL=(3^{K+1}-1)/2$$

为单向广度优先搜索所搜索的节点数，显然，时间复杂性与空间复杂性都与SL成正比。对于双向广度优先搜索，假定两个方向的搜索在第S层相遇（即得到相同节点）而结束

（1≤S≤K）。 这时，各自搜索的节点数分别为：（3^{S+1}-1）/2，（3^{K-S+1}-1）/2，记

DL(S)=(3^{S+1}-1)/2+(3^{K-S+1}-1)/2

为双向广度优先搜索的两个方向搜索在第S层相遇时所搜索的节点数，再假定S取值为1，2，．．．，K的概率是相同的，则双向广度优先搜索的平均搜索的节点数为：

$$DL=\frac{1}{K}\sum_{S=1}^{K}DL(S)=\frac{1}{K}\sum_{S=1}^{K}((3^{S+1}-1)/2+(3^{K-S+1}-1)/2)$$

$$=\frac{1}{2K}\sum_{S=1}^{K}(3^{S+1}+3^{S})-1$$

$$=\frac{3^{K+1}}{K}-\frac{3}{K}-1$$

故DL/SL≈2/K，即双向广度优先搜索的平均搜索的节点数约为单向的2/K倍。一般地，K是个很大的数（本例中K≈120）。这表明，双向搜索远远优于单向搜索。

上面仅就一个简化模型讨论的。在我们的实际对照计算中，双向广度优先搜索的运行时间约为单向的十分之一。

4 核心算法代码及其注释

```
procedure xin01( node4:node_ext;ii,jj,kz,kz3:integer); { 处理单个空格}
  var {略}
  begin {1}
   nodetmp:=node4;{处理队列中的队头节点，即准备扩展的节点}
   with node4 do
   begin {node4}
    i:=ii;j:=jj;
    if arr[i-1,j]=1 then
      begin
        arr[i,j]:=1;  i:=i-1;
        toij_1(1,node4,kz,i,j,kz3);
     end;
    i:=ii;j:=jj; {考虑是否可以向上移动空格}
    if arr[i-1,j] in b1 then
      begin
        arr[i,j]:=arr[i-2,j];
        i:=i-2;
        toij_1(1,node4,kz,i,j,kz3);
      end;
    node4:=nodetmp;
    i:=ii;j:=jj; {考虑是否可以向下移动空格，略}
    i:=ii;j:=jj; {考虑是否可以向左移动空格，略}
    i:=ii;j:=jj; {考虑是否可以向右移动空格，略}
  end; {1}
 procedure search; { 建立队列主过程}
 var {略}
  begin  {2}
  while (qend>qhead)and(qend<range2)and(qend2>qhead2)and(qend2<range2) do
```

```
 { 队列可扩展的条件：两个队列均不空且均不满 }
begin {while}
    tmp:=qhead;
    if (tmp mod 500)=0 then
      writeln('tmp:',qhead:2);{监控运行状况}
    snode:=queue[tmp]^;
    snode.parent:=qhead;
    sd_to_ext(snode,snode7);
    xin01(snode7, snode7.i1, snode7.j1, 1, 1); {处理第一个单个节点}
    xin01(snode7, snode7.i2, snode7.j2, 2, 1); {处理第二个单个节点}
    with snode7 do
      begin{node7}
        if (i1=i2)and(j1+1=j2) then   xin002(snode7,i1,j1,1);
         {处理两个节点同时上下移动}
        if (i1+1=i2) and (j1=j2) then   xin003(snode7,i1,j1,1);
         {处理两个节点同时左右移动}
      end; {node7}
    qhead:=qhead+1;
   {以下略}
   end; {while}
end; {2}
```

5　运行结果及说明

直接运行上述程序所得结果为122步（在PIII 933MH微机上运行大约30秒）。在前面提到得81步的走法中，“一步”是指一个棋子走一次，可能走过2个空格，按这一规则可将122步合并为93步。从理论上讲，广度优先搜索应得到最优的结果，而这里93步却多于81步，原因在于我们是在限定每步只走一个空格的前提下的最优，而不是每步既可走一个空格也可走两个空格的意义下的最优，显然。后者的程序实现会更复杂一些。事实上，利用搜索所得的结果，做少量的调整，便可得到81步的走法。

参考文献：

[1] 林尧瑞，马少平　人工智能导论，清华大学出版社，1989

[2] 北京山科技术开发有限公司　山科益智游戏《华容道》，1997