Computer Science and Application 计算机科学与应用, 2012, 2, 22-25
http://dx.doi.org/10.12677/csa.2012.21005 Published Online March 2012 (http://www.hanspub.org/journal/csa)

A New Algorithm for Travel Route Problem of Horse in Chess*

Dan Liu, Feng Jia

College of Computer and Information Technology, Liaoning Normal University, Dalian
Email: {liudan_dl, jfeng0323}@163.com

Received: Dec. 12th, 2011; revised: Dec. 30th, 2011; accepted: Jan. 14th, 2012

Abstract: The travel route problem of horse in chess was brought up in this paper. An efficient algorithm was proposed based on recursive algorithm of backtracking and bestchoicing. The path number of living nodes is sorted in the algorithm of backtracking to improve the efficiency, and the one who has the fewest path number will be the best choice. Implementation procedure is given in this paper, and simulation results show it is effective.

Keywords: Travel Route of Horse; Backtracking; Bestchoicing

国际象棋中马的周游路线问题新解法*

刘　丹，贾　丰

辽宁师范大学计算机与信息技术学院，大连
Email: {liudan_dl, jfeng0323}@163.com

收稿日期：2011 年 12 月 12 日；修回日期：2011 年 12 月 30 日；录用日期：2012 年 1 月 14 日

摘　要：本文对国际象棋中马的周游路线问题进行了分析，设计了基于回溯法和最优选择法的算法，对回溯过程中不同活结点的通路数量进行排序，并将通路数目最少的活结点作为当前的最优选择，从而进一步提高回溯法的效率。文中给出了算法的具体实现过程，实验结果验证了算法的有效性。

关键词：马的周游路线；回溯法；最优选择法

1. 引言

在实际应用中，当要求解国际象棋中某一位置的马，它是否可能只走 63 步，就走过除起点外的 63 个位置的问题时，通常直接求解是很困难的。原因在于如果直接求解，则无法判断每一步是否为最优通路，以及在遇到错误通路时不能有折回的余地。所以，本文提出回溯法和最优选择法二者相结合的一种解决方法。回溯法的基本思想是对整个已知的空间进行探索，若在当前所处的结点处仍可搜索到新的结点，则称该结点为活结点，反之则为死结点。若遇到死结点，则应返回到最近的一个活结点处。回溯法即以这种工作方式对通路递归的进行搜索[1,2]。然而由于题目中涉及的顶点数较多，而且每个位置可选择的位置数又可分为 8、6、4、3、2 个，这样处理起来十分麻烦，而且采用单一的回溯法通常会对结点进行重复搜索，效率不高[3]。所以我们对回溯过程中不同活结点的通路数量进行排序，并将通路数目最少的活结点作为当前的最优选择，从而进一步提高回溯法的效率。

本文对国际象棋中马的周游路线问题进行了分析，设计了基于回溯法和最优选择法的算法，文中给出了算法的具体实现过程，实验结果验证了算法的有效性。

2. 问题的提出

考虑国际象棋棋盘上某位置的马，它是否可能只

*基金项目：本文研究得到辽宁省教育厅科学研究一般项目(L2011192)，计算机软件新技术国家重点实验室(南京大学)开放课题(KFKT2011B09)资助。

走 63 步，正好走过除起点外的其他 63 个位置各一次？如果有一种这样的走法，则称所走的这条路线为一条马的周游路线。

3. 问题的新解法

3.1. 算法的基本思想

将棋盘以 8 × 8 的数组来表示，并将数组定义成为坐标形式，如图 1 所示。首先确定马的起始位置(x, y)，然后根据马的移动规则搜索下一个活结点，将所有可能的活结点构成集合。利用冒泡排序法对集合中所有活结点构成的通路进行排序，并利用最优选择法确定通路最少的活结点。当遇到死结点时，则按照回溯法的思想让马跳回到最近的一个活结点。算法的流程图如图 2 所示。

一般来说，当马位于位置(x, y)时，可以走到下列 8 个位置之一：(x – 2, y + 1)，(x – 1, y + 2)，(x + 1, y + 2)，(x + 2, y + 1)，(x + 2, y – 1)，(x + 1, y – 2)，(x – 1, y – 2)，(x – 2, y – 1)，如图 3 所示。但是，如果(x, y)靠近棋盘的边缘，上述有些位置可能超出棋盘范围，成为不允许的位置。8 个可能位置用两个一维数组来表示，如图 4 所示，位于(x, y)的马可以走到的新位置是在棋盘范围内的(x + xx[i], y + yy[i])，其中 i = 0,1, …,7。每次在多个可走位置中选择其中一个进行试探，其余未曾试探过的可走位置必须用适当结构妥善管理，以备试探失败时的“回溯”(悔棋)使用。

3.2. 算法的实现

Authentication 函数是按照马的移动规则对所有的新结点进行判断，查看其是否为活结点，如果为活结点则记录其通路的个数。Sort 函数是利用通路数量

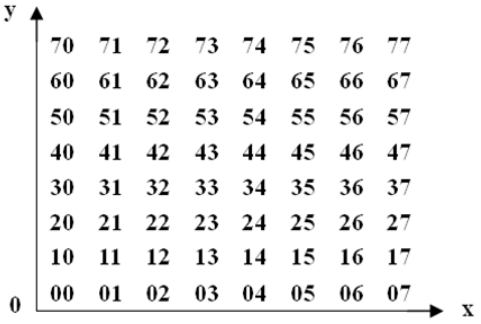

Figure 1. 8 * 8 chess coordinate
图 1. 8 × 8 棋盘坐标

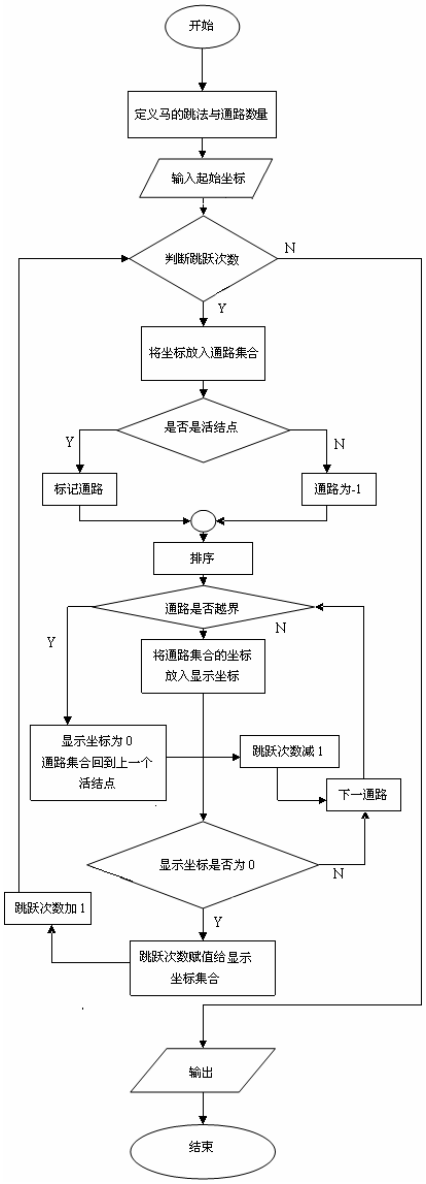

Figure 2. Algorithm flowchart
图 2. 算法流程图

	7		0	
6				1
		马		
5				2
	4		3	

Figure 3. Travel rules of horse
图 3. 马移动规则图

	0	1	2	3	4	5	6	7
xx	1	2	2	1	-1	-2	-2	-1
yy	2	1	-1	-2	-2	-1	1	2

Figure 4. Travel coordinate array of horse
图 4. 马移动位置的坐标数组

对所有的活结点进行冒泡排序，从而选择出最少通路的结点，即最优通路。这里利用了单因素优选法的思想，即如果在试验时，只考虑一个对目标影响最大的因素，其它因素尽量保持不变，则称为单因素问题[4,5]。在这里预先估计最优通路的数量范围，当数量越少，或仅为 1 时，即为最优通路。Query 函数实现回溯的过程，在判断移动次数没有超过包括起点在内的 64 次的前提下，如果遇到活结点则继续按照马的移动规则进行移动，同时记录通路的个数，将排序中通路个数最少的活结点作为下一步的起始结点，递归完成一条马的周游路线；如果遇到死结点则返回上一个活结点处，重新开始判断。

```
void Query(int x,int y)
{int i,xl,yl;//定义工作变量
 while(num<=64)//num 为移动次数
  {for(i=0;i<8;i++)
   {t[i].x=x+xx[i];
    t[i].y=y+yy[i];//将坐标放入通路集合
     Authentication(t[i].x,t[i].y,i);//验证通路数量
   }
   Sort();//排序
   for(i=0;t[i].l<0;i++);//过滤掉死结点
     for(;i<8;i++)
      {xl=t[i].x;
       yl=t[i].y;//将通路集合坐标赋值给工作变量
       if(s[xl][yl]==0)//判断显示坐标是否为 0
       {//将跳跃次数赋值给显示坐标
          s[xl][yl]=num;
          num++;//移动次数增加
  Query(xl,yl);//递归
  if(num>=65)//判断移动次数是否越界
     break;
  s[xl][yl]=0;//将显示坐标置零（此时遇到死结点）
  t[i+1].x-=xx[i+1];
  t[i+1].y-=yy[i+1];//回到上一个活结点
 }
 else
  continue;
 }
if(num>=65)
 break;
num--;//回到上一次移动的活结点处
break;
 }
}
void Sort()//冒泡排序，选出最少通路的通路集合
{int i,j;//定义工作变量
 for(i=0;i<8;i++)
   for(j=i+1;j<8;j++)
     if(t[i].l>t[j].l)
     {t[8]=t[i];  t[i]=t[j];   t[j]=t[8];}
}
void Authentication(int x,int y,int k)//计算通路
{int i,xl,yl,n=0; //定义工作变量
 if(x<0||y<0||x>=8||y>=8||s[x][y]>0)//判断是否死结点
   t[k].l=-1;
  else
    {for(i=0;i<8;i++)
          {xl=x+xx[i];
           yl=y+yy[i];
           if(xl<0||yl<0||xl>=8||yl>=8||s[xl][yl]>0)
              continue;
           else
               n++;
          }
    t[k].l=n;
    }
}
```

4. 实例分析

8 × 8 棋盘大小的马周游路线问题是 n × n 棋盘大小的马周游路线问题的一种具体表现形式，将马放置在任意的空格内，然后让马按规则行走，最终走满 64 个格，其中不能重复跳入同一格内。可指定马的起始位置是(x, y)，然后让马按照移动规则,经行跳跃。由于在接近边缘时，可能会跳出界，所以定义边界条件 if(x < 0||y < 0||x ≥ 8||y ≥ 8||s[x][y] > 0)，t[k].l = –1，即此步不可行。假定当前马的起始位置(x, y) = (3, 6)，即对应图 1 坐标位置的 63，则程序执行结果如图 5 所示。

在实现过程中，通过试探、验证、再试探，最终得到满足要求的解。由于采用了通路数量最小这样的

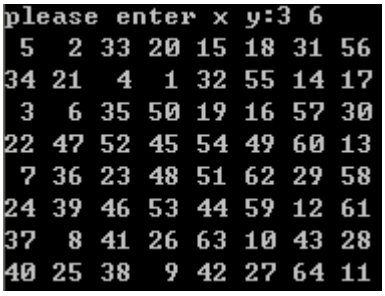

Figure 5. Program execution result
图 5. 程序执行结果

最优通路选择方法，大大减少了一般回溯法中的冗余搜索，保证了算法的效率。由算法的设计可得该算法的时间复杂度为$O\left(n^3\right)$，该时间复杂度优于单一使用回溯法的指数级时间复杂度[6]，并且思路清晰、简单，便于实现编程。

5. 结束语

本文对国际象棋中的马，它是否可能只走 63 步，正好走过除起点外的其他 63 个位置各一次的问题进行了分析，并利用回溯法与最优选择法给出了解题思路，从而减少了运算时间、提高了效率，实验结果验证了其有效性。本文提出的新解法对一般的周游问题同样具有借鉴作用。

参考文献 (References)

[1] 苏德富, 钟诚. 计算机算法设计与分析[M]. 北京: 电子工业出版社, 2001.
[2] 严蔚敏, 吴伟民. 数据结构[M]. 北京: 清华大学出版社, 1999.
[3] 辛玲, 王相海. 国际象棋中马的周游路线问题的递归算法[J]. 计算机工程与设计, 2006, 27(1): 47-49
[4] 杨元法, 庄明. 网络中最短距离的递归算法[J]. 计算机工程, 2005, 31(13): 93-95, 98.
[5] 王晓东. 计算机算法设计与分析[M]. 北京: 电子工业出版社, 2001.
[6] 柏森, 杨晓帆. 求马步图哈密顿圈的最优算法[J]. 计算机工程与科学, 2000, 22(22): 8-11.

