

用 VC++ 实现基于 A* 算法的八数码问题

朱永红, 张燕平

(安徽大学 计算智能与信号处理教育部重点实验室, 安徽 合肥 230039)

摘要:在人工智能领域中,八数码问题一直都是一个游戏难题。介绍了八数码问题,然后在启发式搜索算法上对 A* 算法定义进行了解释,并在其旨在提高搜索效率的方面作了比较详尽的介绍,详细描述了基于图搜索算法的解决此类问题的一种启发式搜索算法——A* 算法。再依据这种算法用可视化编程语言 VC++ 6.0 来实现八数码问题的求解过程,取得了预期的搜索解,提高了搜索效率。

关键词:八数码问题;启发式搜索;A* 算法

中图分类号:TP18

文献标识码:A

文章编号:1673-629X(2006)09-0032-03

Programming for Eight - Figure Puzzle Problem Based
on Algorithm A* with Visual C++

ZHU Yong-hong, ZHANG Yan-ping

(Ministry of Education Key Lab. of Intelligence Computing and Signal Processing,
Anhui University, Hefei 230039, China)

Abstract:Eight - Figure Puzzle problem is always a game puzzle in artificial intelligence. This paper introduces the Eight - Figure Puzzle problem. Then it explains the definition of algorithm A* and makes it clear how to improve the efficiency of search and describe the algorithm A* which is one of heuristic search based on graphsearch. According to this algorithm display the process of exploring the Eight - Figure Puzzle problem by VC++ 6.0, which results in the best solution and improves the efficiency of search.

Key words:Eight - Figure Puzzle problem; heuristic search; algorithm A*

0 引言

八数码问题是人工智能的一个经典的问题。文中通过设计一个基于 A* 算法的状态空间搜索程序,对于给定的初始状态,采用 $h(n) = p(n)$ 表示以每一个将牌与目标位置之间距离的总和作为启发函数的度量,并用可视化编程语言 VC++ 来实现该问题。

实现从初始状态到目标状态的转变,如图 1 所示^[2]。问题的实质就是寻找一个合法的动作序列。



图 1 八数码问题的一个实例

1 问题描述

八数码问题^[1]就是在一个 3×3 的九宫格棋盘上,摆有 8 个将牌,每一个将牌都刻有 1~8 中的某一个数码。棋盘中留有一个空格,允许其周围的某一个将牌向空格移动,这样通过移动将牌就可以不断改变将牌的布局。这种求解的问题是:给定一种初始的将牌布局或结构(称初始状态)和一个目标布局(称目标状态),问如何移动数码,

2 A* 算法

启发式搜索^[3,4]是利用问题拥有启发信息引导搜索,以达到减小搜索范围、降低问题复杂度的目的。在启发式搜索过程中,要对 Open 表进行排序,这就要有一种方法来计算待扩展结点有希望通向目标结点的不同程度,人们总是希望能找到最有可能通向目标结点的待扩展结点优先扩展。一种最常用的方法是定义一个评价函数对各个结点进行计算,其目的就是用来估算出“有希望”的结点。用 f 来标记评价函数,用 $f(n)$ 表示结点 n 的评价函数值,并用 f 来排列等待扩展的结点,然后选择具有最小 f 值的结点作为下一个要扩展的结点。

A* 算法^[4,5]是一种有序搜索算法,其特点在于对评价函数的定义上。这个评价函数 f 使得在任意结点上其函数值 $f(n)$ 能估算出从结点 S 到结点 n 的最小代价路径的

收稿日期:2006-01-03

基金项目:国家自然科学基金资助项目(60475017);教育部博士点基金资助项目(20040357002)

作者简介:朱永红(1981-),男,安徽旌德人,硕士研究生,研究方向为智能计算;张燕平,教授,硕士研究生导师,研究领域为人工神经网络、机器学习、人工智能在金融工程中的应用。

代价与从结点 n 到某一目标结点的最小代价路径的代价的总和,也就是说 $f(n)$ 是约束通过结点 n 的一条最小代价路径的代价的一个估计。再定义一个函数 f^* ,使得在任意一个结点 n 上的函数值 $f^*(n)$ 就是从结点 S 到结点 n 的一条最佳路径的实际代价加上从结点 n 到目标结点的一条最佳路径的代价之和,即

$$f^*(n) = g^*(n) + h^*(n)$$

评价函数 f 是 f^* 的一个估计,这个估计可由下式给出:

$$f(n) = g(n) + h(n)$$

其中 g 是 g^* 的估计; h 是 h^* 的估计。对 $g^*(n)$ 的估计 $g(n)$ 的选择就是搜索树中从 S 到 n 的这段路径的代价,这一代价可以由从 n 到 S 寻找指针时,把遇到的各段路径的代价加起来给出。 $h^*(n)$ 的估计 $h(n)$ 依赖于有关问题的领域的启发信息,于是称作启发函数。在启发函数中,应用的启发信息(问题知识)越多,扩展的结点就越少,这样就能更快地搜索到目标结点。文中运用的是下面这种评价函数:

$$f(n) = d(n) + p(n)$$

其中, $d(n)$ 代表搜索树中结点 n 的深度,根结点深度是 0。启发函数 $p(n)$ 定义为每一个结点与其目标位置之间距离的总和。

3 问题的 VC++ 6.0 实现

3.1 程序界面的设计

本次设计是用 VC 来实现的。程序界面如图 2 所示。

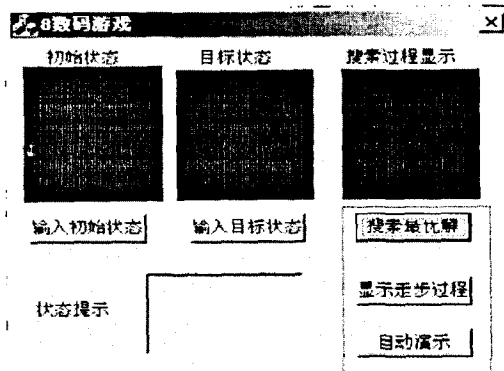


图2 八数码问题示例程序

(1) 状态输入。

可以通过界面上的输入初始状态和输入目标状态的两个按钮来输入初始、目标状态。输入好之后如图 3 所示。

(2) 搜索最优解。

通过点击搜索最优解按钮可以在状态提示栏中显示对于刚输入的初始状态到目标状态需要的步骤。显示如图 4 所示。

(3) 求解步骤显示。

可以通过点击显示步骤按钮来显示每一步是怎样移

动的,也可以通过点击自动演示按钮来显示每一步的走动情况。但是两个区别是后者只需单击一下就会自动显示每一步骤,而前者点击一次显示一步。对于本问题需单击此按钮 5 次。

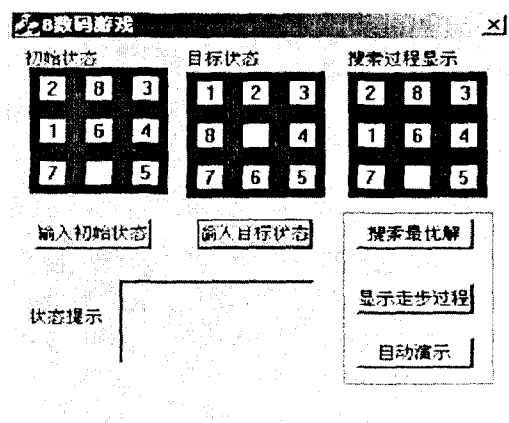


图3 初始、目标状态输入

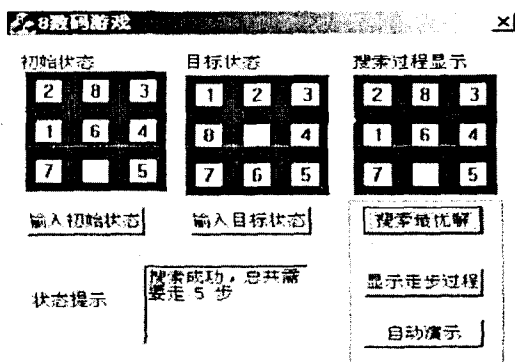


图4 问题求解结果显示

3.2 A* 算法设计

本程序中用到的结构体和类^[5]如下:

```
typedef struct JiuGongState
{
    int curdistance; //当前的距离,从初始节点开始
    int state[3][3]; //存储每个将牌的位置坐标
    struct JiuGongState * prestate; //上一个结点的指针
    struct JiuGongState * nextstate; //下一个结点的指针
} JGState; //定义九宫结构

class CJiuG
{
public:
    JGState StateInit; //初始状态
    JGState StateObj; //目标状态
    JGState StateCur; //当前状态
    CPtrList OpenList; //Open 表
    CPtrList CloseList; //Close 表
    CPtrList ResultList; //保存结果
    JGState * curstep;
public:
    int m_ ndepth; //搜索深度
    CJiuG();
```

```

virtual CJiuG();
bool MoveLeft(JGState * src,JGState * result); //左移
bool MoveRight(JGState * src,JGState * result); //右移
bool MoveUp(JGState * src,JGState * result); //上移
bool MoveDown(JGState * src,JGState * result); //下移
bool Compare(JGState * src1,JGState * src2); //比较两个状态是否相等
int ComputeFn(JGState * cur,JGState * dest); //估价函数的计算,我们采用了 Pn
bool Search(); //用 A* 算法搜索最优解
};

```

本程序的关键是用 A* 算法来搜索最优解,所以程序中的核心部分是 Search()函数的实现。设计思路如下:

首先比较初始状态和目标状态是否相同,如果相同则搜索成功并且退出,不相同则将起始结点加入到 Open 表中去,然后搜索 Open 表中估计值最小的结点。在这里采用的启发函数是 $h(n) = p(n)$,即每一个将牌与其目标位置之间距离的总和,在程序中是 ComputeFn()函数:

```

int CJiuG::ComputeFn(JGState * cur,JGState * dest)
{
    int xcur[9],ycur[9],xdest[9],ydest[9]; //保存 9 个坐标
    int i,j;
    int result=0;
    for(i=0;i<3;i++)
    {
        for(j=0;j<3;j++)
        {
            xcur[cur->state[i][j]]=i;
            ycur[cur->state[i][j]]=j;
            xdest[dest->state[i][j]]=i;
            ydest[dest->state[i][j]]=j;
        }
    }
}

```

(上接第 31 页)

当的优化及处理,可以进一步发挥 SVM 本身处理问题的优越性。

表 1 数据实验结果表

序号	训练集预处理前			训练集预处理后		
	实际结果	预测结果	总耗时(s)	实际结果	预测结果	总耗时(s)
1	+1	+1	302	+1	+1	233
2	+1	+1		+1	+1	
3	+1	+1		+1	+1	
4	+1	+1		+1	+1	
5	+1	+1		+1	+1	
6	-1	+1*		-1	-1	
7	-1	+1*		-1	-1	
8	-1	-1		-1	-1	
9	-1	-1		-1	-1	
10	-1	-1		-1	-1	

4 结束语

将 RS 和 SVM 相结合,利用 RS 对 SVM 处理的训练

```

}
//计算当前状态的每个将牌的与目标状态之间的距离的总和。
for(i=1;i<9;i++)
{
    result=result+abs(xcur[i]-xdest[i])+abs(ycur[i]-ydest[i]);
}
return result;
}

```

再将估价函数最小的结点从 Open 表中删除,加入到 Close 表中去。然后对该结点进行扩展,分 4 种情况(空格左移,右移,上移,下移)讨论,再对 4 种移动所得到的结点重复上述操作过程直到找到目标结点为止。

4 结 论

本程序主要是用 A* 算法来搜索八数码问题的最优解。通过输入大量的初始状态和目标状态发现,在一般情况下都可以找到最优的动作序列,但对某些复杂的初始状态虽能得到正确解却不能完全得到最短的搜索路径。这是有待改进的地方。

参考文献:

- [1] 林尧瑞,马少平.人工智能导论[M].北京:清华大学出版社,1989.
- [2] 马少平,朱小燕,人工智能[M].北京:清华大学出版社,2004.
- [3] 尼尔逊 N J.人工智能原理[M].北京:科学出版社,1983.
- [4] Ansari N,Hou E.用于最优化的计算智能[M].李 军,边肇祺译.北京:清华大学出版社,1999.
- [5] 王万森.人工智能原理及其应用[M].北京:电子工业出版社,2000.

样本进行预处理,可以缩短 SVM 的训练时间,并在不影响 SVM 预测系统预测精度的前提下提高 SVM 预测系统的实时性,为实际预测问题的处理提供了一个很好的解决方案。

参考文献:

- [1] 张 辉,张 浩,陆剑峰.SVM 在数据挖掘中的应用[J].计算机工程,2004,30(6):7-8.
- [2] 张文修,吴伟志,梁吉业,等.粗糙集理论与方法[M].北京:科学出版社,2001.
- [3] Pawlak Z. Rough sets[J]. International Journal of Information and Computer Science,1982,11:241-256.
- [4] 李孟歆,吴成东,夏兴华.粗糙集理论及其应用[J].沈阳建筑工程学院学报,2001,17(4):296-299.
- [5] 邓乃扬,田英杰.数据挖掘中的新方法-支持向量机[M].北京:科学出版社,2004.