人工智能中求解八数码问题算法的实现与分析

张 鸿

(郑州大学 升达经贸管理学院,河南 郑州 451191)

摘 要:针对八数码问题的求解,给出了深度优先搜索、广度优先搜索和启发式搜索(譬如 A* 算法)之间的算法比较,通过实验验证各种算法并得出结论:在通常情况下,采用启发式搜索算法来进行状态空间的搜索更为方便、高效。

关键词:深度优先搜索;广度优先搜索;启发式搜索; A* 算法;八数码问题

中图分类号:TP312 文献标识码:A 文章编号:1672-7800(2009)06-0062-03

0 引言

在人工智能领域,提供的每种问题求解方法,都需要某种对解答的搜索,从提出问题(即初始状态)到问题的解决(即目标状态),有个求解的过程,也就是搜索过程。用于搜索的方法主要有两大类:一类是盲目搜索,另一类是启发式搜索。盲目搜索是指在不具有对待定问题的任何有关信息的条件下,按固定的步骤进行的搜索,如深度优先搜索和广度优先搜索; 启发式搜索是指在搜索中加入了与问题有关的启发性信息,这些信息可以指导搜索朝着最有希望的方向前进, 加速问题的求解过程,并找到最优解,譬如 A* 算法。本文针对上述几种搜索算法,对八数码问题进行了求解,并进行了比较,力求更快速、更高效地找到问题的解答。

1 八数码问题的描述

所谓八数码问题起源于一种游戏: 在一个 3×3 的方阵中放入八个数码 1、2、3、4、5、6、7、8 ,其中一个单元格是空的。将任意摆放的数码盘(称初始状态)逐步摆成某种给定的数码盘的排列(称目标状态) ,如图 1 所示。

图 1 八数码问题的某个初始状态和目标状态

对于以上问题, 我们可以把数码的移动等效成空格的移动。如图 1 的初始排列,数码 7 右移等同于空格左移。那么对于每一个排列,可能的一次数码移动最多只有 4 种,即空格左移、空格右移、空格上移、空格下移。最少有两种(当空格位于方阵的 4 个角上时) 。所以,问题就转换成如何从初始状态开始,使空格经过最小的移动次数最后使排列变成目标状态。

[3] 高尚,杨静宇.群智能算法及其应用[M].北京:中国水利水电出版社,2006.

[4] 赵传信,季一木.粒子群优化算法在 0/1 背包问题的应用[J].微机发展,2005(10).

[5] 孙建中.0/1 背包问题动态规划算法的探讨[J].现代计算机,2005(12).

(责任编辑:杜能钢)

Particle Swarm Optimization about 0/1 Knapsack Problem

Abstract: As a typical combinatorial optimization problem in Operations Research, knapsack problem has a broad background and many different ways to solve. This paper provides a method based on PSO and this method uses some ideas about Genetic Algorithm to apply PSO in knapsack problem and gained better results.

Key Words: Knapsack Problem; PSO

作者简介:张鸿(1975-),女,河南郑州人,硕士,郑州大学教师,研究方向为面向对象技术和数据库技术。

2　八数码问题的求解算法

2.1　盲目搜索

常见的盲目搜索算法有宽度优先搜索和深度优先搜索。

(1)宽度优先搜索算法——扩展当前节点后生成的子节点总是置于 OPEN 表的后端,即 OPEN 表作为排队表使用,先进先出(FIFO),使搜索优先向横广方向发展。

(2)深度优先搜索算法——扩展当前节点后生成的子节点总是置于 OPEN 表的前端,即 OPEN 表作为栈表使用,后进先出(FILO),使搜索优先向纵深方向发展。

2.2　启发式搜索

启发式搜索算法的基本思想是:定义一个评价函数(Evaluation Function) f,对当前的搜索状态进行评估,找出一个最有希望的节点来扩展。

先定义下面几个函数的含义:

$$f^*(n)=g^*(n)+h^*(n) \quad (1)$$

式中 $g^*(n)$表示从初始节点 s 到当前节点 n 的最短路径的耗散值;$h^*(n)$表示从当前节点 n 到目标节点 g 的最短路径的耗散值;$f^*(n)$表示从初始节点 s 经过节点 n 到目标节点 g 的最短路径的耗散值。

评价函数的形式可定义如(2)式所示:

$$f(n)=g(n)+h(n) \quad (2)$$

其中 n 是被评价的当前节点。$f(n)$、$g(n)$和 $h(n)$则分别表示是对 $f^*(n)$、$g^*(n)$和 $h^*(n)$3 个函数值的估计值。

利用评价函数 $f(n)=g(n)+h(n)$来排列 OPEN 表节点顺序的图搜索算法称为算法 A。在 A 算法中,如果对所有的 x,

$$h(x)\leqslant h^*(x) \quad (3)$$

成立,则称 $h(x)$为 $h^*(x)$的下界,它表示某种偏于保守的估计。采用 $h^*(x)$的下界 $h(x)$为启发函数的 A 算法,称为 A* 算法。

针对八数码问题我们可以将其启发函数设计如下:

$$f(n)=d(n)+p(n) \quad (4)$$

其中 $g(n)$根据具体情况设计为 $d(n)$,意为 n 节点的深度(从起始节点到当前节点 n 已经走得步数)

$h(n)$根据具体情况设计为 $p(n)$,其可以有两种设计思路:

(1)放错位的数码个数;

(2)放错位的数码与其正确的位置距离之和。

启发函数中,应用的启发信息(问题知识)愈多,扩展的节点数就越少。其搜索的范围就愈小,搜索速度就越快。显然,(1)中的启发式函数就不够贴切,从而在搜索过程中"不够精确的"地选用了多余节点加以扩展。而(2)中设计的启发函数 $h(n)$更接近于 $h^*(n)$,因为其更接近实际移牌的情况,利用的更多的信息,自然扩展的节点就会更少,搜索效率会更高,后面的程序验证可以说明这一点。

由(1)和(2)两种思路而设计的启发函数,都符合 A* 的条件算法的条件。现仅以思路(2)加以说明:由于实际情况中,一个将牌的移动都是单步进行的,没有交换牌等这样的操作。所以要把所有的不在位的将牌,移动到各自的目标位置上,至少要移动从他们各自的位置到目标位置的距离和这么多次,所以最优路径的耗散值不会比该值小,因此该启发函数 $h(n)$满足 A* 算法的条件。

3　程序实现

本文采用 VC++ 6.0 基于对话框的程序设计,在图形界面上显示出八数码格局并可以随意调整格数码位置。确定初始状态格局后,分别有"深度优先"、"广度优先"、"A* 算法(放错位的数码个数)"、"A* 算法 (放错位的数码与其正确的位置距离之和)"等算法供选择。点击搜索按钮,开始执行搜索,成功或失败后提示, 并显示其生成节点个数与以遍历节点个数。其中在"A* 算法(按不在位的距离和评估)"算法搜索成功后,在树控件中显示,搜索最佳路径的遍历生成树。

限于篇幅本文仅给出启发式搜索算法实现,盲目搜索算法类似可以实现。启发式搜索算法分别用两种启发函数方法实现,其中函数 AX_search()表示启发函数为:放错位的数码个数。函数 AXX_search()表示启发函数为:放错位的数码与其正确的位置距离之和。

其中函数 AXX_search()函数关键实现语句如下:

```
int EightGame::AXX_search()
{
    e->child[n]=new EightGame;//分配子节点空间
    e->child[n]->parent=e;
    e->child[n]->InitNode(m_order,m_emptyGrid);
    e->child[n]->fx_value();//估价函数给对象的属性 f 赋值
    count=open->GetCount ();//count 为当前 open 表元素个数
    for(int i=1;i<count;i++)//循环从 open 第二个(open 表从 0 开始)开始
    {
        POSITION pos=open->FindIndex(i);//pos 结构体赋值
        te1=(EightGame*)open->GetAt(pos);
                    //获得 pos 位置的指针强制转换为//EightGame* 型 te1
        if(te1->f>e->child[n]->f){
            open->InsertBefore(pos,e->child[n++]);//插入
                    break;}
    }//for 结束
    if(count==open->GetCount())
        //如果 open 表的元素个数没有改变, 说明都比//child[n]->f 小,则插入到队尾
    open->AddTail(e->child[n++]);
    m_total++;//扩展子节点完成自增
```

……

return m_total;

}// AXX_search()结束

AX_search()代码结构与此函数相同,只是在 EightGame 对象的 f 赋值时调用的函数不同而已。

4 程序运行结果与分析说明

用 A* 算法(启发函数为思路为放错的数码的个数)实现八数码问题的运行界面如图 2 所示,其它算法运行结果图限于篇幅,此处略去。

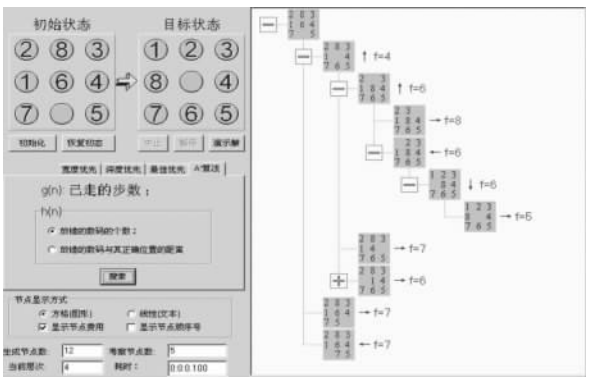

图 2 A* 算法实现八数码问题程序实现界面

针对如图 1 所示的初始状态和目标状态,程序分别使用不同的深度优先搜索、广度优先搜索和启发式搜索算法(两种不同的启发函数)实现八数码问题的任务。表 1 分别反映了这几种算法在解决八数码问题时的运行结果参数:

从表 1 的数据可以看出,当初始状态和目标状态一致时,实验运行结果显示,如果采用深度优先算法,节点有可能一直漫无目的的扩展下去,如果不加层数溢出限制很难在短时间内找出解。而如果采用宽度优先算法,理论上可以找到解,但如果初始节点与目标节点差距过大,在短时间内也不能有效求解,而采用 A* 算法远远优于盲目搜索,并且估价函数的选择更能体现出算法的搜索效率。

表 1 不同算法实现八数码问题运行结果参数对照表

所用算法	共生成节点数	考察节点数(最优路径所需节点数)	搜索到目标需要的层数	耗时(毫秒)
宽度优先搜索算法	35	20	4	1242
深度优先搜索算法	258	143	4	89263
A* 算法:放错的数码个数	12	5	4	100
A* 算法:放错位的数码与其正确的位置距离之和	10	4	4	90

备注:"耗时"只有在相同的计算机软硬件条件下才有参加意义

而对比 A* 算法中两种启发函数,因为使用"放错位的数码与其正确的位置距离之和"作为启发函数比使用"放错的数码个数"作为启发函数,应用的启发信息(问题知识)更多,所以其搜索的范围就愈小,搜索速度就越快,自然扩展的节点数就越少。

5 结束语

通过分析不同搜索算法和不同估价函数对程序效率的影响,可以为实际设计搜索算法和估价函数的选择提供参考依据。搜索算法作为计算机科学的一个重要的研究领域,在当前搜索引擎迅猛发展的环境下,搜索所需要的有效而实用的搜索策略将会越来越得到人们的重视,其研究的重要意义也日趋明显。

参考文献:

[1] 马少平,朱小燕.人工智能[M].北京:清华大学出版社,2004.
[2] 詹志辉.通过八数码问题比较搜索算法的性能[J].计算机工程与设计,2007(6).
[3] 朱福喜.人工智能基础教程[M].北京:清华大学出版社,2004.

(责任编辑:卓 光)

The Solution and Analysis of Eight Puzzle Problem in Artificial Intelligence

Abstract: With using the depth-first search (DFS) and the breadth-first search and heuristic search algorithms (such as A* algorithm) to solve the eight puzzle problem, the performance of these algorithms is analyzed. The results of experiments show the analytic conclusion that in general heuristic search algorithms have more convenient and efficient than blinding algorithms in the states space search.

Key Words: Depth-first Search; Breadth-first Search; Heuristic Search; A* Algorithm; Eight Puzzle Problem