一种改进的广度优先求解华容道问题的方法①

李彦辉　李爱军　(山西财经大学 信息管理学院 山西 太原 030006)

摘　要： “华容道”是中国传统的拼版类游戏。本文通过对华容道求解问题的分析，提出高效且方便的布局表示算法，并在此基础上对广度优先搜索方法进行改进，采用折叠累加产生 HashCode 的方法求解华容道问题。通过实验证明：该改进方法不但具有很好的求解效率，而且性能不会随问题复杂度的增加而骤降。

关键词： 人工智能；广度优先搜索；华容道；HashCode

Method for Hua Rongdao Based on Breadth-First Search

LI Yan-Hui, LI Ai-Jun(Department of Information Management, Shanxi University of Finance and Economics, Taiyuan 030006, China)

Abstract: Hua Rongdao is a very famous puzzled of China.In this paper,through analysis of Hua Rongdao,an efficient and convenient algorithm was used to express the layout.And then we use a Hash algorithm,which HashCode was found based on cumulative fold,to solve the problems.The experiment shows that this method is a very efficient way in finding the answer.

Keywords: artificial intelligence; breadth-first searche; Hua Rongdao; HashCode

1　引言

华容道是中国传统的益智游戏，取材于《三国演义》中《诸葛亮智算华容、关云长义释曹操》一章，以其丰富的益智性和趣味性与七巧板、九连环被合称为“中国古代三大不可思议的游戏；以其变化多端、百玩不厌的特点与魔方、独立钻石棋一起被国外智力专家并称为“智力游戏界的三个不可思议”。

该游戏的棋盘是由 20 个小正方形组成的 4×5 的长方形。共有 10 个棋子，4 个占 1 格的棋子为刘备兵，1 个占 4 个格子的大正方形棋子为曹操。5 个占 2 个格子的棋子为张飞、赵云、马超、关羽和黄忠五虎大将。棋盘上仅有 2 个格子空着。游戏的玩法是通过棋子在这个长方形的棋盘内滑动，在有解的情况下，用最少的步数把曹操移到正下方出口即为胜利。如图 1 所示的即为被称为“横刀立马”的初始布局。按一个棋子走一个空格，或连续走两个空格算一步的规则，日本藤村幸三郎曾在《数理科学》杂志上发表华容道“横刀立马”的最少步法为 85 步。后来清水达雄找出更少的步法为 83 步。美国著名数学家马丁·加德纳又进一步把它减少为 81 步。此后，至今还未曾见到打破这一纪录的报道。而对解题效率的提升，则有数学家指出，此问题单靠数学方法很难求解。我们研究的目的是应用计算机对每种布局快速地找到最优走法或判断它无解。

图 1　横刀立马

① 基金项目:贝叶斯分类器与判别式学习方法研究(60873100)
收稿时间:2010-03-05;收到修改稿时间:2010-04-01

2 游戏求解问题的分析及算法的提出

由于人工智能中比较成熟的盲目搜索算法主要有广度优先算法和深度优先算法。文献[1]采用了改进的非递归深度优先算法对华容道进行求解，但其遍历所得到的首解并不是问题的最优解，需要进一步遍历找出最优解。若问题需要快速地给出一个可行解，相信深度优先算法是一个不错的选择。但本文研究的目的是应用计算机对每种布局快速地找到最优走法或判断它无解，因此本文采用了改进的广度优先算法进行求解。因为从理论上看，通过广度优先搜索找到的第一个解，在解存在的前提下一定是一个搜索步数最少的解，这正好满足华容道求解问题的需要。但是鉴于广度优先搜索普遍存在效率低下的问题，因此着力提高搜索效率成为解决问题的关键。

2.1 相关介绍

2.1.1 广度优先搜索

广度优先搜索表现出来的是一种谨慎，它按照一种同心圆的方式，首先访问所有和初始节点邻接的节点，然后是离它的节点最近的所有未访问节点，依此类推，直到所有与初始节点在一个连通分量中的节点都访问过了为止。

2.1.2 基于链地址法的 Hash 算法

散列法是一种非常高效的基于字典的查找方法。其基本思想是把键分布在一个称为散列表的一维数组 H[0..m-1]中。我们可以通过对每个键计算某些被称为 Hash 函数的预定义函数的值，来完成这种分布。该函数为每个键指定一个称为散列地址的位于 0 到 m-1 之间的整数。显然，如果选择的散列表长度 m 小于键的数量 n，就会遇到碰撞，这是一种两个(或者更多)键被散列表中同一个单元格的现象。

链地址法(如图 6 所示)正是有一种碰撞解决机制的散列算法的版本。在链地址法中，键被存储在依附于散列表单元格上的链表中。每个链表包含着所有散列到该单元格的键。当算出 Hash 函数的值 HashCode 时，先定位到键值为 HashCode 的单元格上的链表，然后查找整个链表，返回找到与否。

2.2 华容道问题的分析

华容道问题具有如下特性：如图 2 所示，棋盘上任意两个形状相同、摆放相同的棋子是“同质”—在计算机看来没有任何区别的。

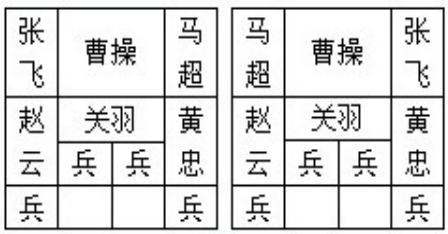

图 2 相同布局

对于问题求解来说，计算机将认为以上两个棋局是完全相同的，也就是说计算机不会记录“人物”信息，只存储“布局”信息。这样做一方面可以减少内存空间的占用，另一方面可以将 UI 与解题过程剥离开，降低耦合，允许各自变化。

基于以上性质，为了描述方便，在本文中作如下约定：“曹操”代表占 2×2 格子的棋子、“横二”代表占 1×2 格子的棋子、“竖二”代表占 2×1 格子的棋子、“兵”代表占 1×1 格子的棋子、“空格”代表棋盘上的空格。

2.3 布局表示算法

在进行广度优先搜索时，一个盘面状态可以理解为一个节点。文献[2]用 5 行 4 列的二维数组来表示一个盘面状态。由于其对于盘面的表示和搜索的操作非常不方便，笔者提出了盘面与 64 位整数相转换的算法来简化搜索树中节点的表示以提高求解效率。

棋盘布局的存储方式为：将棋盘分为 4×5=20 个单元格，如图 3 所示分别标上序号 0~19。在棋子排放上，从棋盘左上角开始，每个棋子的摆放位置都选取最小可用的行、列坐标位置作为摆放位置。每个棋子使用 3 个二进制位来描述，共需要 60 个二进制位，故使用 8 字节(保留 4 个二进制位)表示一个棋盘布局。在.NET 环境下可以使用一个 Int64 进行存储。这样一个布局就可以和一个整数相互转换。

0	1	2	3
4	5	6	7
8	9	10	11
12	13	14	15
16	17	18	19

图 3 棋盘标号

编码规则为：000 – “空格”、001 – “兵”、010 – “竖二”、011 – “横二”、“曹操”的编码为其对应位置序号的 3 个二进制位值。

例如，“曹操”放在了序号为 1 的棋盘格上，那么棋局的物理存储为：00000000-00000000-00000000-00000000-00000000-00000000-00000000-00001000

又如，当“曹操”放在了序号为 1 的棋盘格上，“竖二”放在了序号为 0 的棋盘格上，那么棋局的物理存储为：

0000000-00000000-00000000-00000000-00000000-00000000-00000000-00001010

下图 4 的棋盘布局如果转换成 Int64 的话，就是：

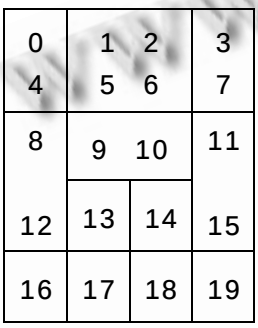

图 4　棋盘布局

对应位置：|19|18|17|16|15|14|13|12|11|10|9|8|7|6|5|4|3|2|1|0

对应的物理存储：0000|100|000|000|100|000|100|100|000|010|000|011|010|000|000|000|000|010|000|001|010

转换成 Int64：

577606461035643914

2.4 折叠累加产生 HashCode

广度优先[3]求解华容道问题，是采用广度优先的树型结构进行计算机搜索，当搜索到符合解的布局时便终止搜索，并从树型结构中回溯出问题求解的全过程。

对于广度优先需要不断地生成子节点，经过实验发现对于 10 棋子的布局来说，广度优先树型结构某层上的最多布局数在 200 到 800 之间。而对于这些节点必须进行合理的存储才不至于造成混乱，因此笔者采用了如下图 5 所示的链表和树型结构相结合的数据结构，它非常便于进行广度优先搜索，因此当我们试图搜索第三步可行解时，只需要顺着第二步的链表依此搜索即可。

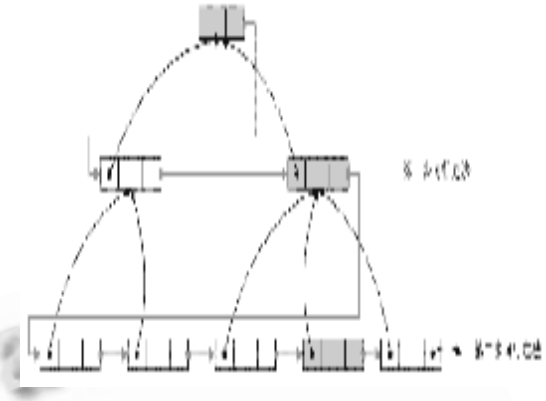

图 5　链表与树型结构的结合

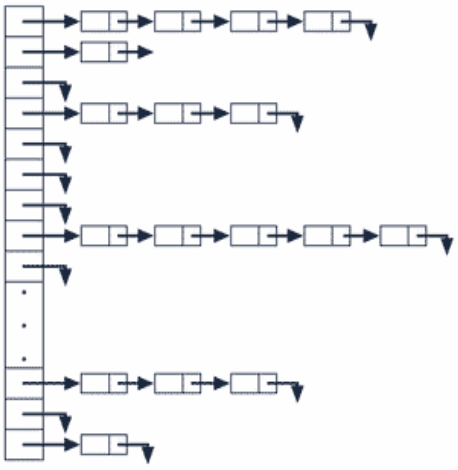

图 6　链地址法

在问题的求解过过程中，要搜索并判断某一布局以前是否出现过，以便及时消除重复布局。这是整个广度优先耗时最多的部分之一，也是广度优先算法能否得到实际应用的关键所在。文献[2]所采用的办法是从头到尾扫描表中的节点以检查某一布局是否出现过，这样在节点数很多的情况下将严重影响效率(这点从本文的实验结果中可以看到)。为此笔者采用了基于链地址法[4]的 Hash 算法作为判断布局是否重复的方法。Hash 算法是一个非常优秀的算法，它的平均时间复杂度理论上为 1。求解中，笔者使用了链地址法实现 HashTable(并没有使用微软提供的 HashTable)。因为尽管从理论上讲，二次探测法的效率比链地址法效率高，但是在华容道的求解过程中，链地址效率要

高很多。另外，HashCode 的计算方法采用了折叠累加的方法，通过对布局 Int64 表示进行折叠累加得到 HashCode。

链地址法如下图 6 所示。

链地址法的数据插入和删除比较简单，如果 HashCode 的计算是高效的，那么在同一个位置的键值就很少，搜索长度也会很短，而且可以有效地节省空间。

本文给出的具体的折叠累加算法为：

第 1 步：输入 Int64 表示的布局值 layout。

第 2 步：设定 14 位的二进制“11111111111111”对应的 Int64 类型的值 mash=16383。

for(int i=0;i<=5;i++)

HashCode+=(int)((layout 逻辑按位与(mash 向左位移 i×14 位))向右位移 i×14 位)

第 3 步：返回 HashCode。

经过实验验证，本算法在一个 HashCode 下最多出现 10 个节点，平均为 3 个节点左右。可见对于缩短搜索长度是非常有效的，进一步提高了求解问题的效率。

3 实验及结果分析

图 7 6 种布局

根据以上方法，笔者在.NET 环境下用 C#语言进行了程序实现。并对华容道的一些布局做了测试。为了与文献[2]提供的数据进行比较，本方法针对文献[2]提供的六种布局如图 7 所示，测试数据如表 1。

表 1 测试数据比较

布局名称	求解时间（秒）		最优解（步）		测试节点数（个）
	文献[2]	本算法	文献[2]	本算法	
闯五关	1	0.047	34	34	1202
水泄不通	14	0. 109	79	79	8837
横刀立马	25	0.188	81	81	11955
小燕出巢	35	0.203	103	103	14468
近在咫尺	80	0.234	98	98	21819
走投无路	3	0.063	无解	无解	2976

本算法在 Intel Core 2 CPU T5470@1.60GHz、1GB 内存、Windows SP3、Visual Studio 2008 SP1 环境下采用 C#语言编译运行，虽然和文献[2]的运行环境不同，但是即使考虑到硬件环境上的差异，在效率上仍有明显的提高。另外从问题的横向对比上看，如图 8 所示。

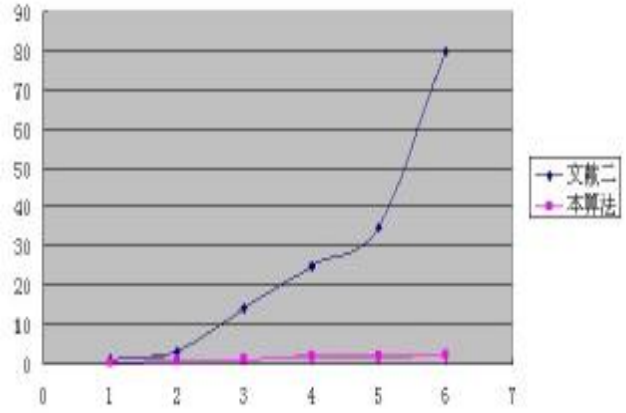

图 8 横向性能比较

横坐标 1、2、3、4、5、6 分别代表上述测试点数由少到多的 6 个问题，为了显示明显，折线图下面

(下转第 194 页)

(上接第 225 页)
的折线为本算法针对每个问题运行时间统一扩大 10 倍所得到。

从上面的折线图可以看出，文献[2]的算法并没能很好地解决广度优先搜索在测试节点个数增加的情况下所带来的程序运行效率的影响。而本算法通过对盘面布局的优化表示并配以相吻合的HashCode产生算法在链表和树型结构相结合的数据结构下对各个问题的求解均是高效且稳定的。

4 结论

广度优先搜索策略是对于求解华容道问题的一种比较好的方法。但是该方法普遍存在效率低下的问题，通过合理的改进和应用能够很大程度上提高搜索效率。实验证明，本文提出的改进方法能够高效而且稳定地对华容道问题进行求解。下一步研究中尝试引入随机的思想来进一步提高算法的求解速率。

参考文献

1 李瑞民,蒋昌俊.“华容道”游戏解法的研究与实现.计算机工程与应用, 2007,43(13):108－110.
2 林宣治.华容道问题最优解的搜索策略.漳州师范学院学报(自然科学版), 2003.11:36－41.
3 严蔚敏,吴伟民.数据结构(C 语言版).清华大学出版社. 1997:169－170.
4 Anany Levitin 著,潘彦译.算法设计与分析基础.第二版.清华大学出版社. 2007:203－205.