博弈树置换表启发式算法研究

焦尚彬,刘 丁
JIAO Shang-bin,LIU Ding

西安理工大学 信息与控制工程研究中心,西安 710048
Xi'an University of Technology,Xi'an 710048,China
E-mail:jiaoshangbin@xaut.edu.cn

JIAO Shang-bin,LIU Ding.Research on translation table heuristic algorithm.Computer Engineering and Applications, 2010,46(6):42-45.

Abstract: Searching is essential for computer-game of board games.And excellent search algorithm may obtain the optimal path by searching few nodes,and improve the competitive level of computer game.This paper sets Chinese chess computer game as background,and on the basis of alpha-beta algorithm,it makes a detailed description about the principle of translation table heuristic algorithm and hash collision,and the replacement schemes of two-level transposition table is proposed,enhancing the efficiency of searing engine.At last,the experiment results verify the effectiveness of the methods.

Key words: computer game;searching tree;translation table heuristic;alpha-beta algorithm

摘 要: 博弈树搜索对于计算机博弈至关重要。优秀的搜索算法通过搜索较少的节点就可以获得最佳路径,从而提高计算机的博弈水平。论文以中国象棋计算机博弈作为背景,在alpha-beta基本搜索算法上,详细阐述了置换表启发算法的原理和哈希冲突,引进了双层置换表的概念及其替换策略,增强了引擎的搜索效率。实验结果表明了该算法的有效性。

关键词: 计算机博弈;博弈树;置换表启发;alpha-beta 算法
DOI: 10.3778/j.issn.1002-8331.2010.06.012 **文章编号:** 1002-8331(2010)06-0042-04 **文献标识码:** A **中图分类号:** TP18

1 引言

计算机博弈的核心是搜索，一盘完整的棋局需要考虑约$45^{90}\approx10^{150}$种局面,因而就目前的电脑硬件水平无法实现"象棋不败算法"[1]。1950年香侬教授(ClaudeShannon)首先提出了"极大-极小算法",奠定了计算机人机博弈的理论基础[2-3]。Bruno在1963年提出了alpha-beta算法[4],1975年Knuth和Moore给出了其数学正确性证明[5]。alpha-beta是所有剪枝算法的基础,但其效率与子树节点搜索的先后顺序密切相关,因此启发算法的目的就是尽可能首先找出最佳着法。其中,历史启发[6]和杀手启发算法[7]都是针对多次搜索的节点排序,具有较高的效率,但没有根据象棋里经常会有重复局面出现的特征来很好利用前面搜索的结果;吃子启发算法[8]根据当前吃对方棋子所能取得的好处排序,符合象棋的特征,但一般局面的吃子着法极少,并且吃子也并不一定就是最佳着法,甚至有可能贪吃一子导致满盘皆输;置换表启发[9]充分利用了象棋里重复局面多次出现的特征,弥补了上述启发算法的不足。

目前,对于单置换表策略主要有深度优先(Deeper Priority)和随时替换(Always Replace)两种。深度优先策略主要基于深度的考虑,忽略了棋局不断演变产生的新棋局信息对以后局势的影响,无法保证棋局信息的实时性,从而降低了置换表的效率,有时甚至引发非法剪枝。而随时替换策略虽然充分考虑了棋局新信息,具有较好的实时性,但却丢掉了拥有较深层数的棋局数据,浪费了资源并增加了搜索时间,效率不如深度优先策略(下文验证实验结果证明了这一结论)。基于以上考虑,利用上述两种置换策略的优点,引入双置换表(Two-level Transposition Table(Two-level TT))策略,并且应用Zobrist哈希技术解决了置换表的存取速度慢及存储空间庞大等问题,分析了置换冲突并给出了风险降低措施。将该策略与alpha-beta搜索算法相结合,实验结果表明,既具有较高的实时性又对已有数据有较好的利用率。

2 alpha-beta基本原理

对于程序而言,考虑对弈双方的每一种着法,就形成了博弈树。一棵深度为3的博弈树如图1所示。

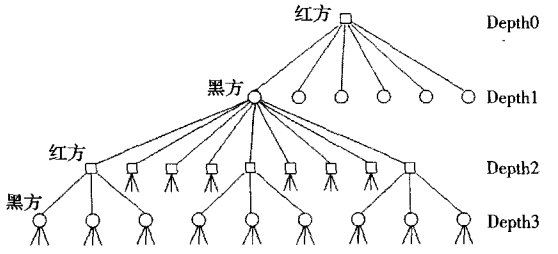
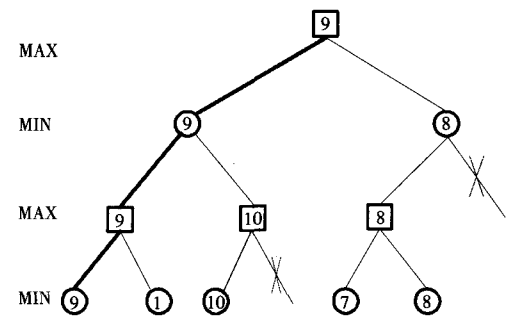


图1 深度为3的博弈树结构图

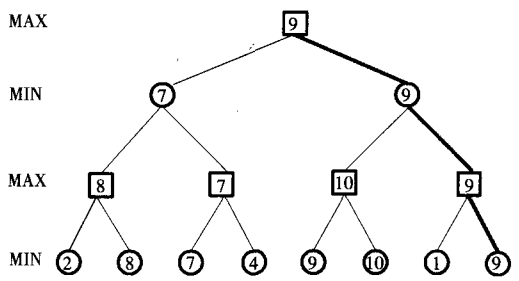
作者简介: 焦尚彬(1974-),男,博士,副教授,主要研究领域为智能控制、状态检测;刘丁(1957-),男,博士,博士生导师,长期从事工业自动化、智能控制理论与应用等方面的研究。

收稿日期: 2008-09-19 **修回日期:** 2008-12-10

博弈程序的基本思想：对有利于本方的棋局给予一个较高的价值分数，不利于本方（有利于另一方）的给予一个较低的价值分数，双方优劣不明显的局面给予一个中间价值分数展开博弈树。对于同一个评价函数而言，若以本方棋局得分与对方棋局得分之差作为评价标准，则本方得分最大即为对方得分最小，本方希望所走着法分值越大，而对方则希望分值越小。因此，alpha-beta 算法通过下界（alpha）和上界（beta）对博弈树的搜索范围进行了划定，当某些子树其值被证明会在上述界限之外（即大于 beta 或小于 alpha），就无法影响整棵博弈树的值，便可进行剪枝。如搜索的顺序是先左节点后右节点，理想情况（搜索节点最少）下 alpha-beta 剪枝过程如图 2(a)所示，最糟糕情况（搜索节点最多）下 alpha-beta 剪枝过程如图 2(b)所示。



(a)理想情况下 α-β 剪枝



(b)最糟糕情况下 α-β 剪枝

图 2　α-β 剪枝搜索示意图

图 2 表明，α-β 剪枝算法的效率与子树节点搜索的先后顺序密切相关。在最理想情况下（极小树），即最左路的分枝就是最佳路径，其生成的节点数目为：

$$N_D=2B^{D/2}-1(D\text{ 为偶数}) \tag{1}$$

$$N_D=B^{(D+1)/2}+B^{(D-1)/2}-1(D\text{ 为奇数}) \tag{2}$$

其中 D 为搜索深度，B 为分枝因子（Branching factor）。而在最糟糕情况下，右节点的得分情况总是好于左节点，搜索过程将不做任何剪枝，此时需要搜索的节点数是 $N_D=B^D$。简单分析可知，最理想情况下 α-β 算法搜索深度为 D 的节点数相当于搜索深度为 $D/2$ 的不做任何剪枝的节点数，由此可见对着法（节点扩展的分枝）排序，即启发算法的研究是搜索的重要课题。

3　置换表启发算法

3.1　置换表的建立

置换表（Translation Table，TT）是基于空间换时间的思想，其基本原理是用一张表把搜索过的信息记录下来，如果将要搜索的某个节点已经有记录，就直接利用记录下来的结果引入到当前的搜索中，即利用置换现象来减少搜索。然而，在象棋的中局阶段，置换现象并不很普遍，因此其主要功效在于启发，即根据置换表来调整搜索次序，属于启发算法的范畴。

TT 结合 alpha-beta 搜索的启发算法基于以下考虑：在 alpha-beta 搜索的过程中，一个节点会出现以下三种情况：(1)Fail High，即节点评估值具体数据并不知道，但其值≥beta。(2)Fail Low，不知道节点评估值具体数据，但其值≤alpha。(3)Exact，则 alpha≤节点评估值≤beta，此值为准确值。在搜索过程中，只有 Exact 类型才可作为当前节点的准确值存入 TT。由于实际对弈过程中，博弈树极其庞大，要搜索的节点接近于无穷，所以在置换表中直接命中的概率非常小，但 Fail High(≥beta)、Fail Low(≤alpha)所对应的边界值仍可帮助对博弈树进行剪枝。同时，考虑置换表在时间和空间的复杂度，实现置换表必须解决以下问题[10]：

(1)查找记录中的节点数据时，速度要非常快，最好是类似于随机存取；

(2)将节点数据放入记录的速度也要非常快；

(3)要能在有限的存储空间内进行。

目前，建立置换表的最有效方法是基于 Zobrist 键值的哈希方法[10-11]。在程序启动的时候，建立一个随机多维数组 ZHash[chessType][position]，其中 chessType 是棋子类型，position 表示棋子位置。基于以上描述，哈希数组的结构如下所示：

```
typedef struct HASHITEM {
    int64 checksum;//哈希值，用以验证表中数据是否要找的局面
    int depth;//该项表示求值时的搜索深度
    //表项值的类型
    enum {exact,lower_bound,upper_bound} entry_type;
    int eval;
    MOVE best;
}HASHE{HASH_TABLE_SIZE};
//定义大小为 HASH_TABLE_SIZE 的哈希数组
```

搜索之前，当前棋盘的 Hash 值便是棋盘上所有存在棋子的坐标上对应 ZHash[chessType][position]的异或之和。这样产生移动后，并不需要重新计算棋盘的 Hash 值，只需将当前 Hash 值：

(1)减去原位置棋子，异或移动棋子原坐标对应 ZHash 值；

(2)修改新位置上的棋子，异或移动棋子新坐标对应 ZHash 值；

(3)如果产生吃子的话，需要异或被吃子在其对应坐标的 ZHash 值；

(4)异或 MoveSideHash，表示改变待走棋一方的颜色。

同时，局面在哈希表中的位置则由式(3)得到：

$$HashIndex/HASH\_TABLE\_SIZE \tag{3}$$

其中，$HashIndex$ 为各棋子哈希值之和 $HASH\_TABLE\_SIZE$，为哈希表大小。由于电脑除法运算速度太慢，当除数是 2^n 时除法可以由右移指令取代，所以 $HASH\_TABLE\_SIZE$ 是一个 2^n 的常量，一般其设定值总是呈倍数增长的，如 4 M、8 M、16 M、32 M 等，但不要超过实际内存的一半。

3.2　哈希冲突分析

简单分析可知，哈希函数是一种“压缩映象”，它把记录的关键字取值很大的数据集中映射到一个范围确定的表中，因此，冲突不可避免。哈希表存在两种冲突：第一种冲突是，不同的局面对应于一个相同的哈希键值。其数学描述为，对于选定的哈希函数 H，$K_i\neq K_j$ 而 $H(K_i)=H(K_j)$，称之为Ⅰ型冲突。当Ⅰ型冲突发生时，会将错误的信息写入置换表，在下次搜索时就会产生错误的搜索路径。解决Ⅰ型冲突的一种有效办法就是检查置换表所指示的着法在当前局面下是否合理，如果合理则有

表1 不同置换策略搜索节点数随置换表的大小变化对比

置换策略	置换表大小						
	16k	32k	64k	128k	256k	1 024k	2M
alpha-beta	6 897 347	6 897 347	6 897 347	6 897 347	6 897 347	6 897 347	6 897 347
alpha-beta+Always Replace	5 733 075	4 968 849	4 461 894	3 938 385	3 543 857	3 237 615	2 762 387
alpha-beta+Deeper Priority	4 744 685	4 184 620	3 946 662	3 516 268	3 043 109	2 732 729	2 407 864
alpha-beta+two-level TT	3 674 217	3 216 233	2 855 502	2 640 304	2 427 866	2 286 471	2 148 524

可能待搜索局面即为置换表所存局面;若该着法对于当前局面不合理,则可以肯定发生了冲突。该方法虽然不能完全避免冲突,但实践证明可以显著降低冲突导致的误搜索。第二种冲突是,不同的局面对应于哈希表中相同的位置,这种冲突被称之为Ⅱ型冲突。因为受实际物理内存的限制,哈希表的大小是有限的,其最大容量也不足以存储所有可能的棋局,冲突在所难免。当这种冲突发生时,就需要决定究竟哪个局面应该继续保存在置换表中,哪个局面应该删除。目前,减少Ⅱ型冲突的最佳办法就是增大哈希索引值的位数,通常设为64位数。实践证明,当将哈希索引值设为64位时,发生冲突的概率几乎可以忽略不计。

3.3 置换策略分析

由于几步以前搜索的结果对本步搜索来说依然具有一定的可信度,因此不必在每次搜索之前都清空置换表,而是保持这些数据作为以后搜索的信息。但数据的可信程度随着棋局的演变不断降低,而置换表的容量是有限的,相对于几近无穷的棋局,当出现重复局面时,需要确定究竟哪些局面该存入置换表中。目前对于单置换表策略,主要有深度优先(Deeper Priority)和随时替换(Always Replace)两种:

(1)深度优先

该策略的基本思想:置换表中原有数据距离叶子节点越远,即深度越深,说明置换表的可信程度越高。在写入哈希表时,如果置换表中已经有了该存储项,且写入数据的搜索深度大于哈希表内已存储数据的深度,则替换掉原有数据;否则说明可信度不高,不进行任何操作。读出时也做相同处理,只有存储数据的深度大于当前搜索深度时,数据才是可信的。由于该策略着重于原有数据的利用,导致了一些最新的信息因为搜索深度的问题而无法保留。

(2)随时替换

该策略的基本思想:置换表中原有数据一般都只有较小的搜索子树,并且原有的搜索深度一般小于后来产生的搜索深度,因而用新产生的棋局信息不做任何判断立即更换置换表中原有数据。该策略具有很好的实时性,但却导致了表中具有较高深度的数据大量丢失。

双置换表(Two-level Transposition Table,Two-level TT)策略结合了以上两种策略的优势,即采用两个置换表分别进行深度优先替换和随时替换策略的搜索。其基本思想:首先基于深度优先的考虑,一般来说深度越深其子树的节点数越多,就更容易显示Two-level TT的优势,减少搜索节点数,并且数据的可靠性越高,因而将深度优先策略作为Two-level TT的第一层;其次,依据局部性原理,最近访问的内容往往在不远的将来会被再次访问的可能性非常大,因而一旦遇到与表中以前所存局面相同时就立即替换,随时替换置换表策略在写入和读出时都不用做深度判断,只要可以剪枝就执行,因此将随时替换策略作为Two-level TT的第二层。然而Two-level TT并不是两个表的简单罗列,其最大的难点在于如何使这两个表有机配合工作,确定什么时候该剪枝及以哪个表中的数据为准进行剪枝。如果配合不好其效率将会大大降低,甚至导致误剪枝等严重后果。另外,两个置换表的同时使用必然增加存储量,从而加大了置换表查找和写入的时间和空间难度等。基于以上问题,置换策略设计如下:

(1)如果待置入节点的深度大于等于Two-level TT第1层节点存储的深度,则将当前第1层存储内容移至第2层,第1层存储写入待置入节点信息;

(2)如果待置入节点的深度小于Two-level TT第1层节点存储的深度,则将待置入节点信息写入第2层存储中。

基于以上置换策略,在应用Two-level TT时,只要其中有一个置换表可以剪枝,则进行剪枝处理。

4 算法验证与结果分析

为了测试Two-level TT的效果,设计了两个测试实验,将两种单置换表替换策略和Two-level TT替换策略启发算法分别结合alpha-beta算法,对比了各种置换策略的搜索效率随置换表的大小及搜索深度的变化关系。表中数据分别表示各种搜索的节点数以及相对于alpha-beta算法的减少量。

(1)实验1

在中局,设定搜索深度为6层,不断增加置换表大小,各种置换策略搜索节点数变化如表1所示。图3表示置换表的大小对不同置换策略效率的影响。

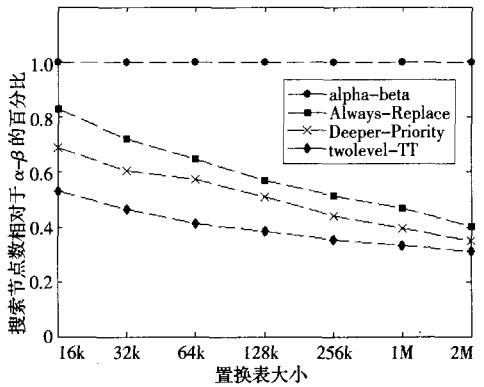


图3 置换表的大小对不出置换策略效率的影响

数据和曲线表明,置换表的加入使alpha-beta搜索的效率明显提高,并且随着置换表的不断增大,置换效果也逐渐增加。在该测试棋局中,当置换表达到2 M时,Two-level TT减少的节点数达到68.8%,并且置换表的增加使得冲突的概率也逐渐下降。

(2)实验2

将置换表的大小设置为16M,不同置换策略随着搜索深度增加其搜索节点数变化如表2所示,图4表示搜索深度对不同置换策略效率的影响。数据和曲线表明,在搜索层数较少的情况下各种置换策略的效果相差不大,但随着层数加深,置换表

表2　不同置换策略搜索节点数随搜索深度变化对比

置换策略	搜索层数					
	2	3	4	5	6	7
alpha-beta	894	15 318	122 891	908 329	6 897 347	48 231 783
alpha-beta+Always Replace	894	12 222	83 971	523 652	3 043 799	18 188 205
alpha-beta+Deeper Priority	894	11 297	77 286	452 530	2 578 918	15 385 939
alpha-beta+Two-level TT	894	10 223	62 773	337 172	2 091 276	11 903 604

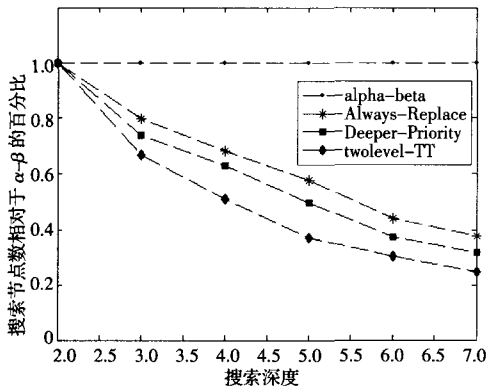


图4　搜索深度对不同置换策略效率的影响

的加入使得搜索效率明显提高，在7层的时候Two-level TT的减少量就已经达到了76.32%，相对于alpha-beta+ Always Replace的68.29%和alpha-beta+ Deeper Priority的71.10%优势明显，这主要是源于搜索层数越深，则重复棋局出现的几率也就越大，Two-level TT引发的剪枝就越多。

5　结论

启发算法对于中国象棋的计算机博弈有着非常重要的作用，好的算法能显著提高搜索引擎的效率，进而提高机器博弈的水平。实验结果表明，Two-level TT既充分利用了以前的搜索结果又很好地保证了棋局的实时性，能够显著提高计算机博弈搜索引擎的效率。事实上，进入残局以后，重复局面出现的几率更大，置换表的优势也将更为明显。

但是，没有任何一种单一的算法能使搜索引擎效率达到最优。各种增强手段之间既可能相互优化，又可能相互冲突。因此如何优化引擎结构以便更快更安全地搜索到最佳结果是计算机博弈领域需要深入研究的课题。

参考文献：

[1] Huang W Q,Song E M.Never lose algorithm in XiangQi[M].Wuhan:Huazhong University of Science and Technology Press,1995.

[2] Plaat A.Research Re:Search and Research[D].Rotterdam:Erasmus University,1996.

[3] Marsland T A.Computer chess and search[M]//Encyclopedia of Artificial Intelligence.Shapiro S.2nd ed.[S.l.]:J Wiley & Sons,1992:224-241.

[4] Brudno A L.Bounds and valuations for shortening the search of estimates[J].Problems of Cybernetics,1963,10:225-241.

[5] Knuth D E,Moore R W.An analysis of alpha-beta pruning[J].Artificial Intelligence,1975,6(4):293-326.

[6] Qian L.The evolution of mulan:Some studies in game-tree pruning and evaluation function in the game amazons[D].University of New Mexico Albuquerque,New Mexico,2003-12.

[7] Sakuta M,Hashimoto T,Nagashima J,et al.Application of the killer-tree heuristic and the lambda-search method to lines of action[J].Information Science,2003,154(3/4):141-155.

[8] 徐心和.中国象棋计算机博弈关键技术分析[J].小型微型计算机系统,2006,27(6):961-969.

[9] Lazar S.Analysis of transposition tables and replacement schemes[R].University of Maryland,1995-12.

[10] Zobrist A.A new hashing method with application for game playing,Technical Report 88[R].Computer Science Department,University of Wisconsin,Madison,1970.

[11] 王小春.PC游戏编程(人机博弈)[M].重庆:重庆大学出版社,2002.

(上接28页)

由图6(a),(c),(e)对直升机俯仰角控制可以看出神经网络PID控制下的系统调节时间最长，系统响应无超调量；模糊PID控制器下的系统调节时间最小，系统响应略有超调量；免疫PID控制下的系统超调量过大，易导致系统不稳定。系统稳态性能方面，神经网络PID控制下的系统无稳态误差；模糊PID控制下的系统存在微量稳态误差；而免疫PID控制下的系统稳态误差非常大，系统控制精度较低。由于免疫PID控制下的系统响应存在很大的超调量和误差，其控制效果无法满足直升机俯仰角的控制指标需求。

由图6(b),(d),(f)对直升机旋转轴控制可以看出神经网络PID控制下的系统调节时间最长，约16秒，系统响应存在超调量；免疫PID控制下的系统调节时间最小，无超调量；模糊PID控制下的系统响应无超调量。系统稳态性能方面，神经网络PID控制下的系统无稳态误差；模糊PID控制下的系统存在微量稳态误差；而免疫PID控制下的系统不存在稳态误差。由于免疫PID控制对旋转轴的控制响应无超调、调节时间最小、无误差，系统输出响应曲线基本还原了阶跃信号的波型，因此完全能够满足直升机旋转轴的最优控制规律和控制指标需求，控制效果相对其他两类控制器改善明显。

根据三种智能PID控制直升机俯仰角和旋转轴的仿真实验和对比研究可知，利用生物免疫机理设计的免疫PID控制器对某类特定的控制模型(旋转轴控制)，其系统响应的动、静态性能指标能够超越模糊PID和神经网络PID控制，这为免疫控制用于其他类型控制无法获得满意控制效果的控制系统，提供了新的设计思路和控制方法。

参考文献：

[1] 薛文涛,吴晓蓓,翟玉强,等.三自由度飞行器模型的神经网络PID控制[J].控制工程,2009,16(2):214-219.

[2] 崔富章.基于BP神经网络的空间飞行器姿态控制研究[D].重庆:重庆大学,2007.

[3] 陈金荣.基于模糊PID的三自由度飞行器模型系统的仿真与控制研究[D].南京:南京理工大学,2006.

[4] 翟玉强.基于三自由度飞行器模型的控制方法研究[D].南京:南京理工大学,2007.

[5] Ishiguro A,Ichikaw A S,Uchikaway A.A gait acquisition of a 6 legged robot using immune networks[C]//IEEE/RSJ/GI International Conference on Intelligent Robots and Systems,Munich,Germany,1994(2):1034-1041.

[6] 谈英姿,沈炯,吕震中.免疫PID控制器在汽温控制系统中的应用研究[J].中国电机工程学报,2002,22(10):148-152.

[7] Mo Hong-wei,Xu Li-fang.Research of a dind of immune fuzzy controller[C]//Proc of 7th International Conference on Mechanics and Automation,2009.