

# “理治棋壮”中国象棋计算机博弈引擎

## 关键算法分析与设计

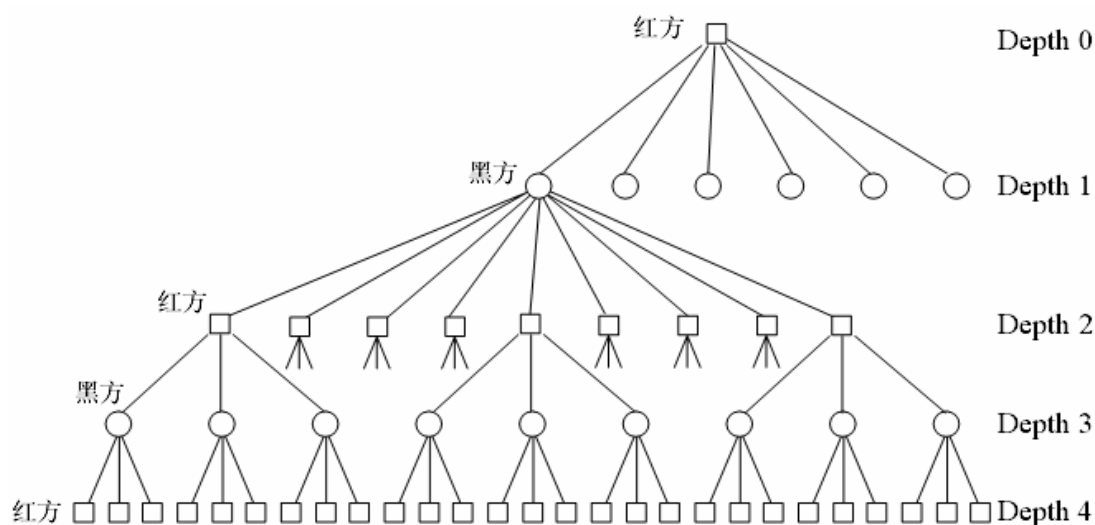
“理治棋壮”中国象棋计算机博弈引擎开发小组

在《程序架构设计与主要算法概述》一文中，我们阐述了中国象棋计算机博弈涉及的主要算法。下面就其中的关键算法进行深入分析，并说明其设计实现方法。

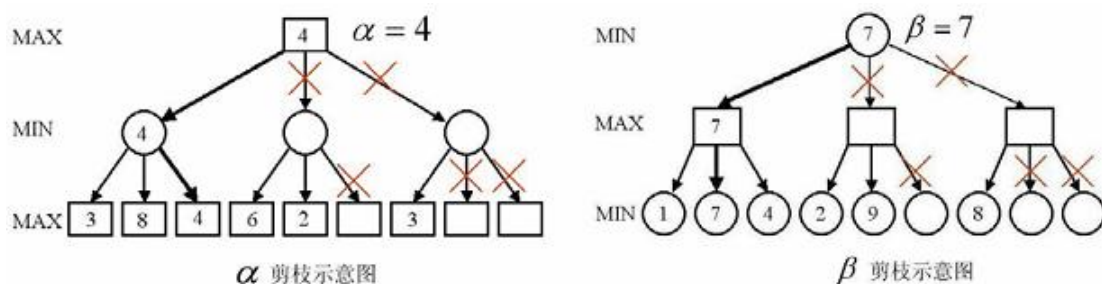
### 一、核心搜索算法

#### 1、Principal Variable Search

搜索算法是计算机博弈程序的核心算法。如何选择适合的搜索算法，配以合理的剪枝条件，是决定搜索效率的关键所在。博弈树不同于一般的搜索树，它是由对弈双方共同产生的一种“变性”搜索树。应对这类问题，香农(Claude Shannon)教授早在 1950 年提出了极大-极小算法 (Minimax Algorithm)。这种算法常以一种形式上的改进——负极大值算法出现：记我方节点值为正，对方节点值为负，双方在每一节点分别选择其子节点中绝对值最大的一个。递归深度优先遍历有限层次的树，找到使起始节点绝对值最大的一个叶子节点，然后回溯找到形成这一叶子节点的第一层子节点，作为最优解。



可以在博弈树深度优先搜索过程中记录 2 个附加值， $\alpha$ ：我方搜索到的最好值，任何比它更小的值就没用了； $\beta$ ：对于对手来说最坏的值。在搜索算法中，如果某个节点的结果小于或等于  $\alpha$ ，那么它就可以抛弃；如果某个着法的结果大于或等于  $\beta$ ，那么整个结点就作废了，因为对手不希望走到这个局面。如果某个节点值大于  $\alpha$  但小于  $\beta$ ，那么这个节点就是可以考虑走。这便是所谓的  $\alpha - \beta$  剪枝搜索。



由  $\alpha$  和  $\beta$  可以形成一个节点预选窗口。如何能够快速得到一个尽可能小而又尽可能准确的窗口？有不少窗口搜索算法被设计出来解决这个问题，如：Aspiration Search、Principal Variable Search、Tolerance Search 等。目前大多数国际象棋与中国象棋的算法核心青睐速度快而不会出现错误剪枝的 Principal Variable Search，它的原理是第一个分枝以完整窗口搜索，产生一个解  $v$ ，后继分枝则以一个极小窗口  $(v, v+1)$  搜索之，旨在建立高效的、极小的搜索树。本质上，极小窗口搜索依赖于后继节点的值对于前驱节点值微小变化的猜测。如果猜测被证明不准确，就要以  $(v+1, \beta)$  为窗口重新进行搜索。

## 2、迭代深化搜索

博弈树的庞大是可以想象的。对于中国象棋，如果按照每一步平均有 45 种可行着法，每局棋平均走 90 步，那从开始局面展开到分出胜负，则要考虑的局面种数比地球上的原子数目还要多，无法完全计算，因此只能对有限层次展开分析。那么应该搜索多少层为佳呢？如果设定一个搜索截止层数，则搜索的时间会随局面上可行着法数量的变化产生显著变化。一个更好的方案是限定搜索截止时间，让程序在有限的时间内尽可能深地搜索。用伪代码表示为：

```
depth = 0;
while (time < limit)
{
    depth++;
    method = search(depth);
}
board.move(method);
```

迭代深化搜索本身也有一些优化措施，例如后一次搜索可以利用前一次搜索的结果调整节点顺序，优先搜索前一次搜索中最优节点的子节点。如果在更少的步数内直接使对方走投无路，应该停止深入搜索，直接走这一步。

## 3、静态搜索

给定深度的博弈树搜索，叶子节点的评估值并不能准确地反映变化剧烈的局面的情况，这称为“水平线效应”。例如某个叶子节点上红炮吃掉了黑马，但是下一步红马会被黑车吃掉。如果在此叶子节点调用评估函数，返回值肯定对红方有利，但会引发无用的兑子。因此有必要加入静态局面搜索，将叶子节点继续展开一直延伸到相对静态（没有吃子）的局面。

静态搜索实现起来比较简单，直接用静态搜索函数替代叶子节点的评估函数即可。在实际应用中多采用选择性延伸，即只对有重要威胁的叶子节点（将军、唯一着法、兑子等）进行展开，忽略次要的非静态叶子节点。

#### 4、循环检测

中国象棋规则中禁止连续三回合以上的循环着法，比如“长将”和“长不变”要判负或判和。如果仅使用普通的搜索和剪枝规则，电脑很可能会死守一个对其有利的局面不放，不断地用循环的着法与对手周旋，最终被判负或判和。因此我们有必要在搜索中记录一个节点所有上级节点的情况，如果它与某个上级节点完全相同，则应避免这个着法（但不可以就此剪枝，因为有时不得不因为只有一条路可走而走两个回合的循环着法）。在具体实现时，不可能记录一个节点（即局面）的全部信息，那样既浪费空间，又难以高效地查找和匹配。通常采用节点的哈希数进行记录和查找匹配。记录的长度应与搜索深度相同。

综合以上算法，我们实现了博弈树搜索模块。请参考源文件 `searcher.cpp`。

## 二、局面评估函数

局面评估就是给棋局打分，识别棋局的好坏，以便在博弈树搜索时作为路径选择条件寻找最佳着法。可行的办法就是对局面进行量化，通过数值评判棋局的好坏。由于评估需要大量的象棋知识，仁者见仁，智者见智，评估函数的设计便成为计算机博弈中最为人性化的部分。一般评估函数要综合考虑以下几个方面：1、棋子价值，2、棋子位置，3、棋子灵活度，4、棋子配合，5、将帅安全等。

1、棋子价值评估值考虑的是某个棋子的存在对一个局面的固有价值，其中将帅为无穷大——它们坚决不可失去。而其它棋子的价值很大程度上来源于人们的经验认识，如车 > 马 > 兵。棋子价值与棋子周围情况完全无关，在与棋子位置评估、棋子灵活度评估共同使用时，其作用往往被后者掩盖，只有“将帅为无穷大”这一点有后者不可替代的意义。

2、棋子位置评估值反映的是同一个棋子处在棋盘的不同位置上攻击力的不同。它可以促使我方摆出有利攻击的局面，同时在搜索算法的配合下，防范对方有利攻击的局面。例如下图所示兵的价值表，兵在过河后、尤其是逼近九宫后价值倍增。总体而言，理论和实践表明：棋子位置评估值在各项评估值中权重是最大的，仅有棋子位置评估值的程序也能够马马虎虎地下棋了。但棋子位置评估对具体一个局面，特别是残局的适应性相对较弱。

9	9	9	11	13	11	9	9	9
29	39	59	74	79	74	59	39	29
29	39	54	64	64	64	54	39	29
29	36	39	49	51	49	39	36	29
19	27	31	44	49	44	31	27	19
7	0	13	0	16	0	13	0	7
7	0	7	0	15	0	7	0	7
0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0

3、棋子灵活度评估记录所有棋子在特定局面下可行着法的总数；棋子配合评估对连环马、马后炮等经典配合打法额外加分；将帅安全评估根据将帅受保护或受威胁的情形加分或减分。相对于针对整个棋盘进行静态局面分析的棋子价值评估和棋子位置评估，后三者虽然也可以通过静态局面评估获得，但这样一来计算量过大，因此通常将它们放在着法生成的同时动态计算，只计算走棋一方的棋子灵活度及所走的那个棋子对棋子配合、将帅安全的影响。由于局面变化的连续性，这种局部动态计算很大程度上可以反映某一着法对局面整体的影响。这三者在残局时发挥的作用较大。因为残局变化多端，子力位置不再为确定值，将军和应将很大程度上要依靠经验的棋子配合打法。

以上各种评估方法得出的评估值加权求和后返回给博弈树搜索模块。其中方法的权值也不应为定值，而是要随局势（开局、中局、残局）的变化动态调整。在开局时加强棋子位置评估以便占据有利位置；在残局时加强棋子配合评估以便快速将军。我们对其实现参见源文件 `evaluation.cpp`。

### 三、其它重要算法

#### 1、哈希表示法

博弈搜索的历史着法检测和开局库检索等方面涉及对棋盘的模式匹配，如果一个一个棋子去匹配，显然太费时，而且保存一个局面的全部信息占据空间较大。解决这个问题的方法是用哈希数保存和检索局面。我们采用博弈算法中经典的 Zobrist 键值表示法，对每个棋子位于每个位置的情况（共 32 个棋子×90 个位置，不排除不可能的位置）用一个 64 位的数（8 个字节）表示。一个局面上所有棋子的 Zobrist 键值之异或就定义为这种局面的哈希表示。理论表明这种哈希表示法冲突概率极低，不用考虑冲突处理。

Zobrist 键值可以通过编写小程序随机生成，将其保存在一个文件（`Zobrist.bin`）中，每 8 个字节为一个 Zobrist 键值。键值文件在引擎加载时读入内存，以后使用时直接调用。在一个局面上添删或移动棋子时，只需将目标棋子在起点和终点的 Zobrist 键值分别异或到原局面的哈希数即可（基于异或运算的“开关”作用：异或奇次改变原数，异或偶次恢复原数）。

对 Zobrist 键值的读取参见源文件 `hashnum.cpp`，局面哈希数的生成参见源文件 `board.cpp`。

#### 2、开局库检索

中国象棋开局的几个回合内，双方各自展开子力，占据棋盘有利位置。如果采用上述通用的博弈树搜索方法往往会因为局部很小的利益而忽视全局的发展，走出贻笑大方的开局。象棋有着经过千锤百炼的经典开局着法，如果将它们存储在数据文件中，在开局时用查询取代局面评估和博弈树搜索，就可以使我们快速出动子力，占据有利位置，同时防止被对手开局的“怪着”整昏。

我们的开局库的来源是《象棋百科全书》网站上收录的 8000 多局全国象棋大赛棋谱，使用该网站提供的小程序将棋谱中的开局提取，按出现的频率和最终的胜率排序，然后使用

自己编写的小程序将上面总结出来的文本格式的开局信息转为我们程序专用的二进制格式开局库文件 (OpenBook.bin)。开局库文件中每 13 个字节为一段, 分别用 8 个字节表示局面 (哈希表示法), 1 个字节表示轮到谁下棋 (0x00 为红, 0x01 为黑), 4 个字节表示应对的着法 (用代表起止位置的棋盘坐标字符串表示, 如 “h0i2” )。

对开局库的读取参见源文件 openbook.cpp, 开局库检索参见源文件 searcher.cpp。

## 附：标准化数据表示方法

1、中国象棋通用引擎协议 (UCCI)：负责博弈引擎与象棋界面程序的通信, 参见：  
[http://www.elephantbase.net/protocol/cchess\\_ucci.htm](http://www.elephantbase.net/protocol/cchess_ucci.htm)。

2、福斯夫·爱德华兹记号法 (FEN)：用一个字符串表示一个局面的所有信息, 参见：  
[http://www.elephantbase.net/protocol/cchess\\_fen.htm](http://www.elephantbase.net/protocol/cchess_fen.htm)。

3、坐标格式 (ICCS)：用代表起止位置的棋盘坐标字符串表示一个着法, 参见：  
[http://www.elephantbase.net/protocol/cchess\\_move.htm](http://www.elephantbase.net/protocol/cchess_move.htm)。

## 参考文献：

- [1] 徐心和、王骄, 中国象棋计算机博弈关键技术分析, 小型微型计算机系统, Vol. 27, No. 6, 2006, pp961-969
- [2] 王小春, PC 游戏编程 (人机博弈), 重庆大学出版社, 2002
- [3] 黄晨, 解剖大象的眼睛——中国象棋程序设计探索, 象棋百科全书网站 (<http://www.elephantbase.net>), 2005
- [4] Bruce Moreland, Min-Max Search - An obvious place to start, Bruce Moreland 个人网站 (<http://www.seanet.com/~brucemo/>), 2001

---

北京理工大学“理治棋壮”中国象棋计算机博弈引擎开发小组版权所有 (2006.7~2006.8)

项目主管：	林 健 (北京理工大学计算机科学技术学院)
指导教师：	黄 鸿 (北京理工大学信息科学技术学院自动控制系统工程研究所)
技术顾问：	赵陈翔 (北京理工大学软件学院)
开发人员：	林 健 (北京理工大学计算机科学技术学院)
	高 然 (北京理工大学计算机科学技术学院)
	应张彬 (北京理工大学软件学院)
	武 斌 (北京理工大学软件学院)

联系方式：  
[lj@linjian.cn](mailto:lj@linjian.cn) (林健)  
[honghuang@bit.edu.cn](mailto:honghuang@bit.edu.cn) (黄鸿)