# 中国象棋 Alpha-Beta 搜索算法的研究与改进*

岳金朋 冯 速
(北京师范大学信息科学与技术学院, 100875, 北京)

**摘要** 针对中国象棋机器博弈中较为高效的 Alpha-Beta 搜索算法进行研究,以提升其效率.依据 Alpha-Beta 搜索算法的效率与子节点扩展的排列顺序高度相关的事实及中国象棋自身的特点,本研究从优化着法的排列顺序入手,设计出了启发能力较强的着法排列方案,并进一步提出了扩大窗口的内部迭代加深算法对上述着法排列方案进行修正,从而使着法顺序得到了进一步的优化.实验数据证明,本研究明显提升了 Alpha-Beta 算法的效率.

**关键词** 中国象棋; Alpha-Beta 剪枝; 着法顺序; 内部迭代加深; 扩大窗口

机器博弈是人工智能研究的重要分支,人类对机器博弈的研究衍生了大量的研究成果.Alpha-Beta 搜索算法是机器博弈领域中最为重要的算法之一.本文在研制中国象棋博弈程序的过程中,设计了一个较为优化的着法生成顺序,对于同一层数可以花费更少的搜索时间,因此棋力更高.在此基础上,提出了一种扩大窗口的内部迭代加深算法,对着法生成顺序做了进一步的优化,从而使 Alpha-Beta 算法的剪枝效率有了较大幅度的提高.在着法生成顺序中,本文还提出了静态评价启发技术.

## 1 极大极小算法(minimax algorithm)

极大极小算法始终站在博弈一方的立场上给棋局估值,有利于这一方的棋局给予一个较高的价值分数,不利于这一方(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则给予一个中间价值分数.在这一方行棋的时候,选择价值极大的子节点走棋,另一方行棋则选择价值极小的子节点走棋.这就是一个极大极小过程.为表述方便,我们将实行极大值搜索的一方称为 max 方,另一方称为 min 方.Shannon[1] 在 1950 年首先提出了极大极小算法,从而奠定了计算机博弈的理论基础.

在极大极小过程中,我们总要检查哪一方取极大值而哪一方又要取极小值,以执行不同的动作.1975 年 Knuth 等[2] 提出的负极大值算法(negamax)消除了两方面的差别,使算法简洁优雅.它的核心思想在于:父节点的值是各子节点的负数的极大值.使用负极大值方法,博弈双方都取极大值.负极大值搜索算法仅仅是一种更好的形式,其原理和极大极小算法完全等效.

## 2 Alpha-Beta 搜索算法

**2.1 Alpha-Beta 搜索算法简介** 在极大极小搜索的过程中,存在着 2 种明显的冗余现象[3].第 1 种现象是极大值冗余.在图 1a 中,节点 A 的值应是节点 B 和节点 C 的值中之较大者.现在已知节点 B 的值大于节点 D 的值.由于节点 C 的值应是它的诸子节点的值中之极小者,此极小值一定小于等于节点 D 的值,因此亦一定小于节点 B 的值,这表明,继续搜索节点 C 的其他诸子节点 E,F,…已没有意义,它们不能做任何贡献,于是把以节点 C 为根的子树全部剪去.这种优化称为 Alpha 剪枝.在图 1b 是与极大值冗余对偶的现象,称为极小值冗余.节点 A 的值应是节点 B 和节点 C 的值中之较小者.现在已知节点 B 的值小于节点 D 的值.由于节点 C 的值应是它的诸子节点的值中之极大者,此极大值一定大于等于节点 D 的值,因此也大于节点 B 的值,这表明,继续搜索节点 C 的其他诸子节点已没有意义,并可以把以节点 C 为根的子树全部剪去,这种优化称为 Beta 剪枝.

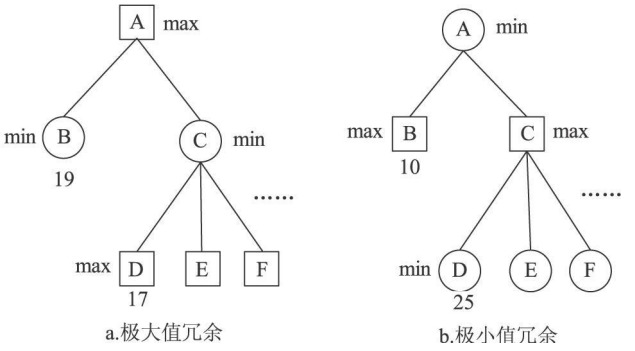


图 1 Alpha 与 Beta 剪枝示意

* 国家自然科学基金资助项目(60273015)
收稿日期:2008-11-25

把 Alpha-Beta 剪枝应用到极大极小算法中，就形成了 Alpha-Beta 搜索算法.

在搜索过程中，max 方节点的当前最优值被称为 α 值，min 方节点的当前最优值被称为 β 值. 在搜索开始时，α 值为 $-\infty$，β 值为 $+\infty$，在搜索过程中，max 节点使 α 值递增，min 节点则使 β 值递减，两者构成一个区间 $[\alpha,\beta]$，这个区间被称为窗口[4-5]. 窗口的大小表示当前节点值得搜索的子节点的价值取值范围，向下搜索的过程就是缩小窗口的过程，最终的最优值将落在这个窗口中. 一旦 max 节点得到其子节点的返回值大于 β 值或 min 节点得到其子节点的返回值小于 α 值，则发生剪枝. 具体算法如下所示：

1) Alpha-Beta 算法，入口参数为 $(P,\alpha,\beta,d)$，其中 $P$ 是节点，$[\alpha,\beta]$ 是初始窗口，d 是搜索深度；

2) 若 $P$ 为叶节点，则取静态估计值返回；

3) 产生 $P$ 的所有合理着法 $m_i$，$1\leqslant i\leqslant n$；

4) $v=-\infty$；

5) $i=1$；

6) 由着法 $m_i$ 生成子节点 $P_i$；

7) $t=-$Alpha-Beta$(P_i,-\beta,-\alpha,d-1)$（递归调用下一层）；

8) 若 $t>v$，则 $v=t$，否则，转 11)；

9) 若 $v>\alpha$，则 $\alpha=v$；

10) 若 $v\geqslant\beta$，则取 $v$ 值返回；

11) $i=i+1$；

12) 若 $i\leqslant n$，则转 6)；

13) 取 $v$ 值返回；

该算法使用了负极大值原理，算法只检查是否出现 β 剪枝，而不需检查是否有 α 剪枝.

在 Alpha-Beta 搜索中，有 3 种不同类型的评价值[6]：高出边界型(fail high)，由于发生了剪枝，只知道返回评分至少是 β，不知道具体值；低出边界型(fail low)，由于没找到较好着法，只知道返回评分最多是 α，不知道具体值；精确型，即返回评分在 α 和 β 之间.

2.2 **Alpha-Beta 搜索算法效率分析**　我们将搜索树平均分枝因子数记作 $b$，搜索深度记作 $d$，那么采用极大极小算法搜索的节点数为

$$1+b+b^2+b^3+\cdots+b^d=b^d\,\frac{1-1/b^d}{1-1/b}\approx b^d.$$

1975 年，Knuth 等[3] 证明了，在节点排列最理想的情形下，使用 Alpha-Beta 剪枝生成的节点数目为

$$N_d=2b^{d/2}-1 \quad (d\text{ 为偶数})$$

$$N_d=b^{(d+1)/2}+b^{(d-1)/2}-1 \quad (d\text{ 为奇数})$$

这个数字大约是极大极小算法搜索节点数的平方根的 2 倍左右. 那么根据公式 $b^{d/2}=(b^{1/2})^d$，我们得出：如果着法顺序的排列最为理想，那么在同样的情况下，我们可以搜索原来深度（即不用 Alpha-Beta 剪枝时的深度）的 2 倍.

由于 Alpha-Beta 剪枝与节点的排列顺序高度相关，寻找有效手段将候选着法排列调整为剪枝效率更高的顺序就显得尤为重要了.

## 3　着法排列顺序的优化

因为 Alpha-Beta 剪枝对总体着法顺序非常敏感，所以采用启发式方法对其进行优化是提高算法效率的关键所在. 本节根据国际象棋中已有的启发式方法，结合中国象棋的实际情况，提出一个较好的着法排列方案，并通过实验进行验证. 在该方案中，我们使用置换表启发、静态评价启发及动态启发等技术.

3.1 **置换表启发(transposition table)**　在搜索进行中，经常会在不同的路径上遇到相同的棋局，这样的子树就没有必要重复搜索，把子树根节点的评价值及其类型、子树的最好着法和取得这个值的深度信息保存在一个称为置换表[7-8] 的数据结构中，下次遇到时直接运用即可.

设现要对某局面进行 $d$ 层搜索，窗口是 $[\alpha,\beta]$，而发现该局面在置换表中已存在，其评价值为 v，类型为 t，且搜索过 d′ 层. 当 $d'\geqslant d$ 时，如果 $t$ 为精确型，便可直接返回 $v$ 而代替搜索；如果 $t$ 为高出边界型且 $v\geqslant\beta$，便可进行剪枝. 否则要进行重新搜索以保证所取得数据的准确程度，此时置换表中保存的最佳着法给我们一定的启发，它为搜索提供一个较优的着法，该着法应该排在最前面优先搜索，这使总体着法排序得到一定程度的优化，置换表最主要的作用就是它对着法顺序的启发[9].

3.2 **静态评价启发(static evaluation heuristic)**　静态评价启发是本文提出的一种虽简单但很有效的启发方法，该方法主要用来优化吃子着法的顺序.

对吃子着法进行静态评价启发，就是找出表面上占有较大优势的吃子着法. 如果我们用 $V$ 表示攻击子的价值，用 $W$ 表示被吃子的价值，那么某个吃子着法的价值 $U$ 可表示为：

$$U=\begin{cases}V-W, & (\text{被吃子有保护})\\ W. & (\text{被吃子无保护})\end{cases}$$

当吃子着法的值 $U>0$ 时为表面上能得到获得很大利益的着法，这样的着法往往是好的着法；而 $U=0$ 可能是等价值的换子，这样的着法也值得一试；而 $U<0$ 的着法往往是吃亏的. 因此我们可将吃子着法依据 $U$ 进行排序，并对 $U\geqslant 0$ 的着法优先进行搜索.

对非吃子着法也可进行静态评价启发，例如，在中

国象棋中,我们可以考虑首先生成车的着法,最后生成帅(将)的着法,往往是很有效的;而在生成车的着法时,首先生成车向前走的着法,然后生成车向后走的着法,也能起到一定的启发作用,而这也和我们下棋时的思考方式十分接近.在中局,我们可以采用车、炮、马、兵(卒)、仕(士)、相(象)、帅(将)的顺序.

3.3 **动态启发(dynamic heuristic)** 相对来说,对于中国象棋中的某一局面,$U\geqslant 0$ 的吃子着法是很少的,大多数着法都是 $U<0$ 的吃子着法和非吃子的着法.对于这些着法仅用3.2节所述的静态评价启发是远远不够的,还要根据搜索中积累的信息进行动态启发.

国际象棋的经验表明,历史启发(history heuristic)[10]和杀手启发(killer heuristic)[11]是很好的动态启发方法.

**3.3.1** *历史启发* 历史启发的思想是:搜索树中某个节点上的一个好的着法,对于其他节点可能也是好的.所谓好的着法即可以引发剪枝的着法或者是其兄弟节点中最好的着法[10].一个这样的着法,一经遇到,就给其历史得分一个相应的增量,使其具有更高的优先被搜索的权利.我们对非吃子着法和表面上不能立刻得到实惠的吃子着法根据其历史得分进行排序,就能获得较佳的着法顺序.由于着法的历史得分随搜索而改变,对节点的排列也会随之动态改变.

**3.3.2** *杀手启发* 杀手启发是历史启发的特例.它是把同一层中引发剪枝最多的着法称为杀手,当下次搜索到同一层次,如果杀手是合法走步的话,就优先搜索杀手.根据国际象棋的经验,杀手产生剪枝的可能性极大,我们在中国象棋里也吸取了这个经验.

3.4 **着法生成顺序** 应用本文3.2节所提出的静态评价启发,结合置换表启发和动态启发,本文提出了一个较好的着法生成顺序,以取代以往以棋子为主键的排列顺序.如图2所示:

使用启发式方法对着法顺序进行优化后,能大大减少搜索的节点数,实验结果如表1所示(测试数据选自文献[12]中有代表性的20个中局盘面),其中优化前的着法顺序指的是以棋子为主键的排列顺序.从表中可以看出,随着搜索深度的增加,优化的着法顺序所起的效用越来越大,这也验证了剪枝的效率与着法顺序是高度相关的.

值得指出的是,在国际象棋中,总是把杀手排在前面优先搜索[11],然而本文通过实验发现,对于中国象棋,将杀手放在静态评价$\geqslant 0$的吃子着法之后,将获得更高的剪枝效率,表2对比了2种方法搜索访问的平均节点数(测试数据为同上的20个中局盘面),从表2可见,这种调整对于中国象棋还是很有效的,体现出中国象棋与国际象棋的不同规则对着法的影响.

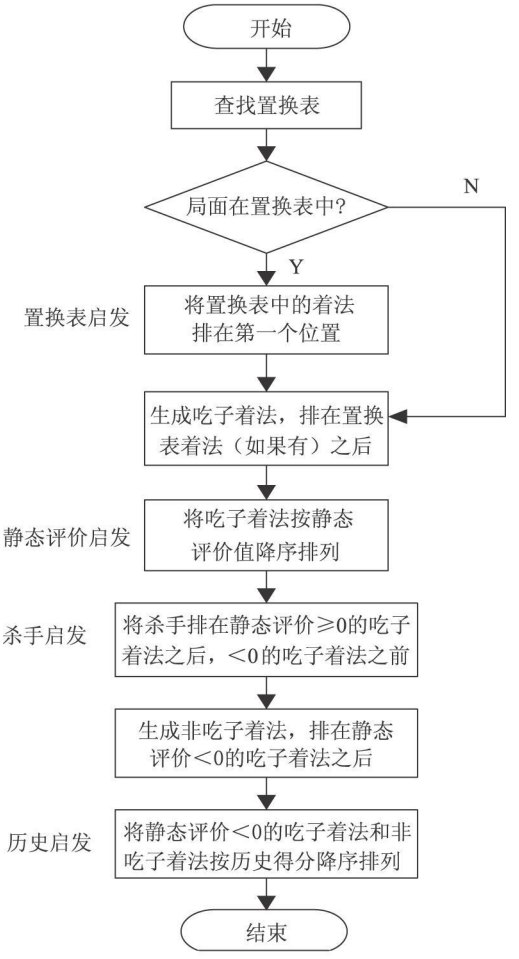


**图2 着法生成顺序**

**表1 不同着法顺序搜索访问的平均节点数对比**

| 搜索深度 | 搜索访问的平均节点数 | | 降低程度/% |
|---|---|---|---|
| | 优化前的着法顺序 | 优化后的着法顺序 | |
| 1 | 40 | 40 | 0 |
| 2 | 670 | 450 | −33 |
| 3 | 5 198 | 2 766 | −47 |
| 4 | 31 408 | 10 882 | −65 |
| 5 | 263 397 | 65 516 | −75 |
| 6 | 1 422 290 | 240 260 | −83 |
| 7 | 11 400 735 | 1 257 174 | −88 |

**表2 杀手排在吃子着法之前与之后搜索访问的平均节点数对比**

| 搜索深度 | 搜索访问的平均节点数 | | 降低程度/% |
|---|---|---|---|
| | 杀手排在吃子着法之前 | 杀手排在吃子着法之后 | |
| 1 | 40 | 40 | 0 |
| 2 | 502 | 450 | −10 |
| 3 | 3 400 | 2 766 | −18 |
| 4 | 14 115 | 10 882 | −23 |
| 5 | 87 275 | 65 516 | −25 |
| 6 | 332 417 | 240 260 | −28 |
| 7 | 1 798 713 | 1 257 174 | −30 |

## 4　内部迭代加深

在 3.1 节曾经指出置换表可为搜索提供一个较优的着法，从而优化了总体着法顺序. 可是，如果查找置换表没有命中，我们如何在一开始就得到一个较优的着法呢？这时就可以利用内部迭代加深[13]的办法.

**4.1　内部迭代加深优化着法顺序**　内部迭代加深是指当对一个节点进行深度为 $d$ 的搜索时，先对该节点做一次深度为 $d-r$ 的搜索，一般来说，$d>3$，$r=2$. 利用 $d-r$ 的浅层搜索可得到一个较优的着法，这个着法将在深层搜索中放在第一个位置被优先尝试，同时，由于浅层搜索将在置换表、历史得分表和杀手表中保存下了有价值的信息，这将使深度为 $d$ 的深层搜索获得更高的剪枝效率.

Knuth 等[2]的实验表明利用 Alpha-Beta 剪枝搜索 $D$ 层所需时间大约是 $D-1$ 层所需时间的 $\sqrt{B}$ 倍，其中 $B$ 为平均分支因子数. 如果中国象棋取 $B=64$，每多搜一层就会花上原先的 8 倍时间. 所以，内部迭代加深中深度为 $d-r$ 的搜索大约是深度为 $d$ 时的 $1/(\sqrt{B})^r=1/8^r$，如果 $r=2$，这个数仅为 1/64. 所以，浅层搜索花掉的代价其实非常小，但却对深层搜索的着法顺序产生了重要指导.

**4.2　扩大窗口的内部迭代加深**　本文通过实践，发现内部迭代加深算法可能会经常发生一个问题：那就是深度为 $d-r$ 的浅层搜索的返回值并未落在窗口 $[\alpha,\beta]$ 的范围之内或高于窗口上界 $\beta$ 发生剪枝，而是低于窗口的下界 $\alpha$，这种低出边界的搜索将导致得到的启发信息不准确. 本文针对这一问题进行了修正，将节点的窗口范围扩大为 $[-\infty,\beta]$，形成了扩大窗口的内部迭代加深（internal iterative deepening with enlarged window，简称IIDEW）算法. 具体算法如下所示：

1) IIDEW 算法，入口参数为 $(P,\alpha,\beta,d,l,S)$，其中 $P$ 是节点，$[\alpha,\beta]$ 是初始窗口，d 是最大搜索深度，l 是当前节点到根节点的距离，S 是一个保存最佳走法的数组；

2) 若 $P$ 为叶节点，则取静态估计值返回；

3) $t=$IIDEW$(P,\alpha,\beta,d-2,l,S)$（内部迭代加深）；

4) 若 $t>\alpha$，转 6)；

5) $t=$IIDEW$(P,-\infty,\beta,d-2,l,S)$（扩大窗口，进行低出边界的修正）；

6) 产生 $P$ 的所有合理着法 $m_i$，并将 $S[1]$ 中的走法排在第 1 个位置；

7) $v=-\infty$；

8) $i=1$；

9) 由着法 $m_i$ 生成子节点 $P_i$；

10) $t=$IIDEW$(P_i,-\beta,-\alpha,d-1,l+1,S)$（递归调用下一层）；

11) 若 $t>v$，则 $v=t$，否则，转 14)；

12) 若 $v>\alpha$，则 $\alpha=v$，$S[1]=m_i$；

13) 若 $v\geqslant\beta$，则取 $v$ 值返回；

14) $i=i+1$；

15) 若 $i\leqslant n$，则转 9)；

16) 取 $v$ 值返回；

在IIDEW 算法中，3)、4)、5) 和 6) 行是对 Alpha-Beta 搜索算法的扩展，其余各行与前述的 Alpha-Beta 搜索算法相同. 扩大窗口的内部迭代加深算法使浅层的启发更加准确，从而对深层搜索做出更加准确的指导.

表 3 对比了扩大窗口的内部迭代加深和普通的内部迭代加深搜索访问的平均节点数（测试数据为同上的 20 个中局盘面），从表中可以看出，扩大窗口的内部迭代加深在第 9 层时能将搜索效率提高 10%左右，并且随深度增加而逐渐提高.

**表 3　普通与扩大窗口内部迭代加深搜索访问的平均节点数对比**

| 搜索深度 | 搜索访问的平均节点数 | | 降低程度/% |
|---|---|---|---|
| | 内部迭代加深+Alpha-Beta 剪枝 | 扩大窗口的内部迭代加深+Alpha-Beta 剪枝 | |
| 1 | 40 | 40 | 0 |
| 2 | 450 | 450 | 0 |
| 3 | 1 734 | 1 734 | 0 |
| 4 | 6 405 | 6 534 | 2 |
| 5 | 43 822 | 42 945 | −2 |
| 6 | 151 590 | 144 010 | −5 |
| 7 | 765 856 | 712 246 | −7 |
| 8 | 2 472 578 | 2 250 045 | −9 |
| 9 | 15 662 623 | 14 096 360 | −10 |

经扩大窗口的内部迭代加深修正后，着法生成顺序如图 3 所示.

## 5　结论

针对中国象棋中的着法排列，本文将置换表着法、静态评价较优的着法和杀手着法排在前面，其余着法按历史得分依次降序排在后面，从而得到了一个较好的着法生成顺序，从实验数据可以看出上述优化在同样的时间内使搜索时访问的平均节点数降低了一个数量级. 然后，通过内部迭代加深算法及其扩大窗口的改进形式，对着法生成顺序做了进一步的优化，并通过实验数据发现扩大窗口的内部迭代加深算法能使剪枝效率有 10%左右的提升.

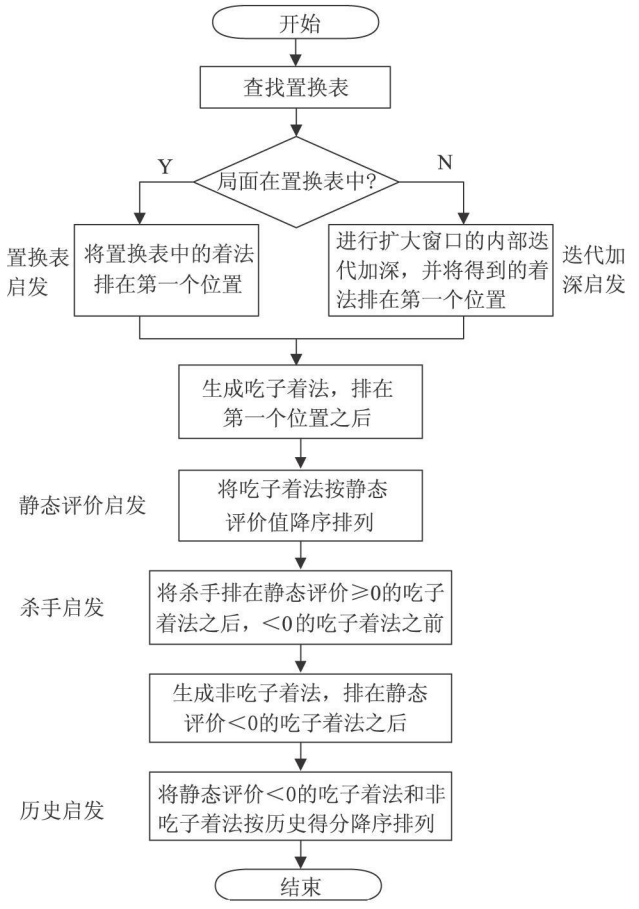


图3 经扩大窗口的内部迭代加深调整后的着法生成顺序

## 6 参考文献

[1] Shannon C E. Programming a computer for playing chess [J]. Philosophical Magazine, 1950,41:256

[2] Knuth D E, Moore R W. An analysis of Alpha-Beta pruning[J]. Artificial Intelligence, 1975,6:293

[3] 王小春. PC游戏编程[M]. 重庆:重庆大学出版社, 2002:102

[4] 陆汝钤. 人工智能(上)[M]. 北京:科学出版社,1995:390

[5] 徐心和,王骄. 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

[6] Eppstein D. Hashing and move ordering [EB/OL]. [2008-11-08]. http://www.ics.uci.edu/~eppstein/180a/970424.html

[7] Zobrist A. A new hashing method with application for game playing[J]. ICCA Journal,1990,13(2):69

[8] Moreland B. Transposition Table[EB/OL].[2008-11-08]. https://chessprogramming.wikispaces.com/Transposition+Table

[9] Breuker D M, Uiterwijk J W H M, Van Den Herik H J. Replacement schemes for transposition tables[J]. ICCA Journal, 1994, 17(4):183

[10] Schaeffer J. The history heuristic and Alpha-Beta search enhancements in practice[J]. IEEE Transactions on Pattern Analysis and Machine Intelligence,1989,11:1203

[11] Akl S G, Newborn M M. The principal continuation and the killer heuristic[C] //Proceeding of ACM National Conference. Seattle:ACM,1977:466

[12] 黄少龙. 象棋中局精妙战法[M]. 北京:金盾出版社, 2005:78

[13] Frayn C. Computer chess programming theory[EB/OL]. [2008-11-08]. http://www.frayn.net/beowulf/theory.html

# IMPROVEMENT ON ALPHA-BETA SEARCH ALGORITHM IN CHINESE CHESS

YUE Jinpeng FENG Su

(College of Information Science and Technology, Beijing Normal University, 100875, Beijing, China)

**Abstract** Alpha-Beta search algorithm in computerized Chinese chess is investigated. Based on characteristics of Chinese chess and the fact that algorithm efficiency is highly affected by move ordering, a move ordering scheme that achieved higher heuristic effect was designed, and proposed new algorithm internal iterative deepening with enlarged window, to revise move ordering scheme. With this algorithm, move ordering has been further optimized. Experimental data showed markedly enhanced efficiency and performance of Alpha-Beta search algorithm.

**Key words** Chinese chess; Alpha-Beta pruning; move ordering; internal iterative deepening; enlarged window