博弈树搜索算法在中国象棋中的应用①

Application of Game Tree Search Algorithm in Computer Chinese Chess

岳金朋　冯　速　(北京师范大学 信息科学与技术学院 北京 100875)

摘　要： 针对中国象棋博弈中较为高效的α－β剪枝算法进行研究，以提升其效率。依据α－β剪枝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，从优化着法的排列顺序入手，设计出启发能力较强的着法排列方案，并进一步提出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进一步的优化。实验数据表明，提出的方法可以明显提升α－β剪枝的效率。

关键词： 中国象棋　α－β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。α－β剪枝是机器博弈搜索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α－β剪枝的效率有了更大幅度的提高。在着法生成顺序中，本文还提出了静态评价启发技术。本文所使用的α－β剪枝的概念及相关的博弈树搜索算法在文献[1]中有较详尽的介绍。

1　着法排列顺序的优化

α－β剪枝对着法顺序非常敏感[2]，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。本节参考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。

1.1 置换表启发

在搜索中，经常会在不同的路径上遇到相同的棋局，这样的子树没有必要重复搜索，把最优着法、分值、深度和节点类型等信息保存在一个称为置换表(transposition table)[3,4]的数据结构中，再次遇到时直接运用即可。

假设对某局面进行 d 层搜索，窗口是[α,β]，而发现该局面在置换表中已存在，其评价值为 v，类型为 t，深度为 d'。当 d'≥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 $v \geq \beta$，便可进行剪枝。否则进行重新搜索以保证所取得数据的准确，此时置换表中保存的最佳着法给了我们一定的启发，它为搜索提供一个较优着法，该着法应排在前面优先搜索，这使总体着法顺序得到一定程度的优化，置换表的一个主要作用是它对着法顺序的启发[5,6]。

1.2 静态评价启发

静态评价启发(static evaluation heuristic)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，就是要找出表面上占有较大优势的吃子着法。中国象棋的主要攻击手段就是吃子，首先通过检测吃子来寻找最优着法往往会产生较好的效果。如果我们用 V 表示攻击子的价值，用 W 表示被吃子的价值(各个棋子的价值如表 1 所示)，那么某个吃子着法的价值 U 可表示为：

$$U = \begin{cases} V - W & \text{（被吃子有保护）} \\ W & \text{（被吃子无保护）} \end{cases} \quad (1)$$

① 基金项目:国家自然科学基金(60273015)
收稿时间:2009-01-09

表 1　棋子的交换价值

帅(将)	车	马	炮	仕(士)	相(象)	兵(卒)
5	4	3	3	1	1	2

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；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≥0 的着法优先进行搜索。

1.3 动态启发

相对来说，对于中国象棋中的某一局面，U≥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 1.2 节所述的静态评价启发是不够的，还要进行动态启发(dynamic heuristic)。树搜索的过程，实际也是信息积累的过程。对这些信息进行挖掘，可以得到我们所需要的启发信息。

国际象棋的经验表明，历史启发(history heuristic)[7]和杀手启发(killer heuristic)[8]都是很好的动态启发方法。历史启发的思想是：搜索树中某个节点上的好着法，对于其他节点可能也是好的。所谓好着法是指可以引发剪枝的着法，或者是其兄弟节点中最好的着法[7]。一经遇到这样的着法，算法就给其历史得分一个相应的增量，使其具有更高的优先搜索权，这个增量通常为 d2(d 为当前节点需要搜索的深度)或 2d。具体地说，就是设立一个 90×90 的数组，红方和黑方的着法都记录在这个数组中，前一个指标代表起始格，后一个指标代表目标格；或者设立一个 14×90 的数组，红方着法和黑方着法分别记录，前一个指标代表兵种，后一个指标代表目标格。历史启发是从全局信息中获取优先准则的一种方法。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。

杀手启发实际上是历史启发的特例。它是把同一层中，引发剪枝最多的着法称为杀手，当下次搜索时，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很大，中国象棋也可以吸取这个经验。

1.4 着法生成顺序

应用 1.2 节提出的方法，本节给出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 1 所示：

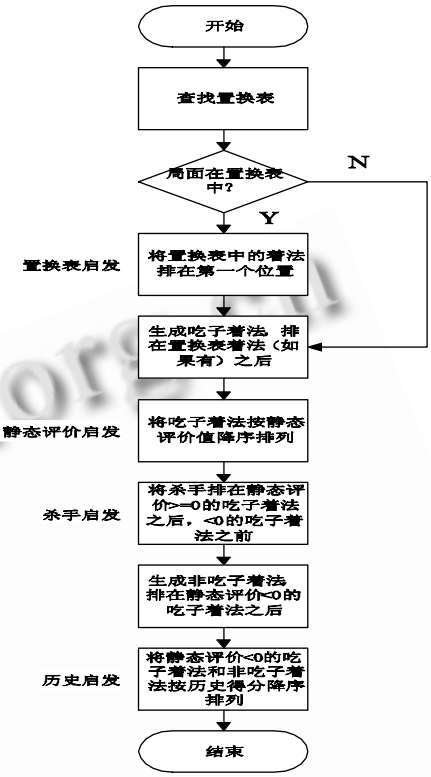

图 1　着法生成顺序

使用启发式方法对着法顺序进行优化可以大大减少搜索的节点数，实验结果如表 2 所示(测试数据选自文献[9]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。

表 2　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	5198	2766	-47
4	31408	10882	-65
5	263397	65516	-75
6	1422290	240260	-83
7	11400735	1257174	-88

值得指出的是，本文对杀手的位置进行了大量实验，发现将杀手放在静态评价≥0 的吃子着法之后将获得更高的剪枝效率。表 3 对比了将杀手排在吃子着法之前与之后搜索时访问的平均节点数(测试数据为上述的 20 个中局盘面)。从表中可以看出，杀手位置的调整对于搜索效率的提升是较为明显的。

表 3　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	3400	2766	-18
4	14115	10882	-23
5	87275	65516	-25
6	332417	240260	-28
7	1798713	1257174	-30

2　内部迭代加深

在 1.1 节曾经指出，置换表可为搜索提供一个较优的着法，从而优化总体着法顺序。然而，如果查找置换表没有命中，我们如何一开始就能得到一个较优的着法呢？这时可以利用内部迭代加深(internal iterative deepening)[10,11]的办法。

2.1　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 的浅层搜索可以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优先尝试，同时，由于浅层搜索将在置换表、历史表和杀手表中保存下有价值的信息，这将使深度为 d 的深层搜索获得更高的剪枝效率。

Knuth 和 Moore 的实验[2]表明利用 α－β 剪枝搜索 d 层所需时间大约是 d－1 层所需时间的 $B^{1/2}$ 倍，其中 B 为平均分支因子数。如果中国象棋取 B＝40，每多搜一层就会花上原先的 6 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 1/6r，如果 r＝2，则浅层搜索所花的时间仅为深层搜索的 1/36，其代价较小，但能够对深层搜索的着法顺序产生重要指导。

2.2　扩大窗口的内部迭代加深

通过实践，本文发现内部迭代加深算法可能会产生一个问题：那就是深度为 d－r 的浅层搜索的返回值并未落在窗口[α，β]的范围之内，而且也没有因为高于窗口上界 β 而发生剪枝，而是低于窗口的下界 α，这种低出边界的搜索将导致浅层的启发信息不准确。本文针对这一问题进行改进，将节点的窗口范围扩大为[－∞，β]，进行低出边界的修正，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。如下所示：

1) 入口参数为(α，β，d)，[α，β]是初始窗口，d 是搜索深度，函数名为 AlphaBeta;

2) 若 d＝0，则取静态估计值返回;

3) HashMove＝0;

4) ProbeHash(α，β，d，HashMove); // 查找置换表

5) 若 HashMove≠0，转 10);

6) Score＝AlphaBeta (α，β，d－2);

7) 若 Score＞α，转 9);

8) Score＝AlphaBeta (－∞，β，d－2);

9) ProbeHash(α，β，d，HashMove); // 查找置换表

10) 产生所有合理着法 mi，1≤i≤n，并将 HashMove 排在第一个位置;

11) Best＝－∞，HashFlag＝“fail low”;

12) i＝1;

13) 执行着法 mi;

14) Score＝－AlphaBeta (－β，－α，d－1);

15) 撤销着法 mi

16) 若 Score＞Best，则 Best＝Score，否则，转 19);

17) 若 Best＞α，则 α＝Best，HashFlag＝“exact”;

18) 若 Best≥β，则 HashFlag＝“fail high”，转 21);

19) i＝i＋1;

20) 若 i≤n，则转 13);

21) RecordHash(HashFlag，Best，d，m_i); // 存入置换表

22) 取 Best 值返回;

加入扩大窗口的内部迭代加深修正后，着法生成

顺序如图 2 所示：

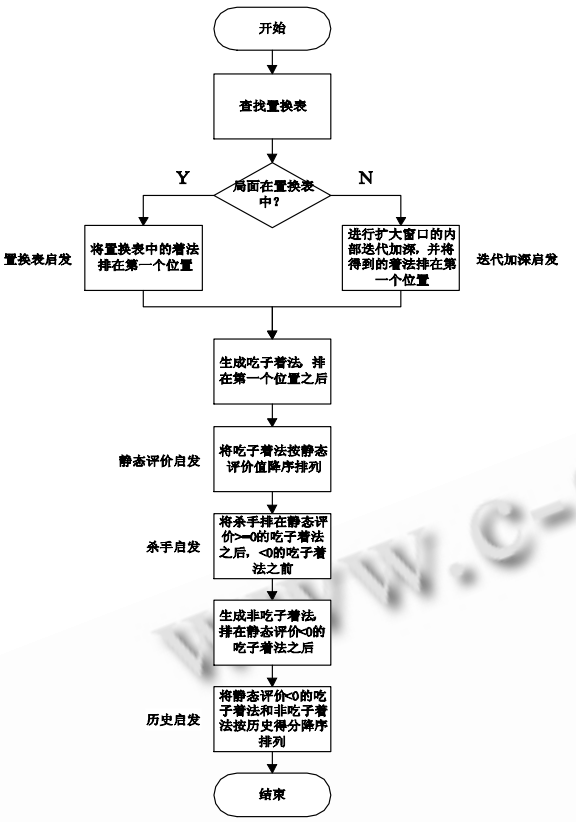

图 2　经扩大窗口的内部迭代加深调整后的着法生成顺序

表 4 对比了普通的和扩大窗口的内部迭代加深搜索访问的平均节点数(测试数据同上述 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 层时将搜索效率提高 10%左右，并且随深度增加而逐渐提高。

3　结论

针对中国象棋中的着法排列问题，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。实验数据表明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。实验数据表明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。因此，本文使用的着法生成顺序对提高中国象棋博弈程序的效率有一定的作用。

表 4　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+α-β 剪枝	扩大窗口的内部迭代加深+α-β 剪枝	
1	40	40	0
2	450	450	0
3	1734	1734	0
4	6405	6534	2
5	43822	42945	-2
6	151590	144010	-5
7	765856	712246	-7
8	2472578	2250045	-9
9	15662623	14096360	-10

参考文献

1 岳金朋,冯速.博弈树搜索算法概述.计算机系统应用, 2009,18(9):203－207.

2 Knuth DE, Moore RW. An analysis of Alpha-Beta pruning. Artificial Intelligence, 1975,6(4):293－326.

3 Zobrist A. A new hashing method with application for game playing. ICCA Journal, 1990,13(2):69－73.

4 Moreland B.Transposition table.2004/3. https://chess-programming.wikispaces.com/Transposition+Table.

5 Breuker DM, Uiterwijk JWHM, van den. Herik HJ. Replacement schemes for transposition tables. ICCA Journal, 1994,17(4):183－193.

6 Breuker DM, Uiterwijk JWHM, Herik HJ van den. Replacement schemes and two-level tables. ICCA Journal, 1994,19(3):175－180.

7 Schaeffer J. The history heuristic and Alpha-Beta search enhancements in practice. IEEE Transactions on Pattern Analysis and Machine Intelligence, 1989,11: 1203－1212.

8 Akl SG, Newborn MM. The principal continuation and the killer heuristic. ACM Proceedings of ACM National Conference. Seattle: ACM, 1977:466－473.

9 黄少龙.象棋中局精妙战法.北京:金盾出版社, 2005: 78－150.

10 Frayn C. Computer chess programming theory. 2006. http://www.frayn.net/beowulf/theory.html

11 Moreland B. Interative deepening. 2004,3.

12 http://www.seanet.com/~brucemo/topics/iterative.htm.