中国象棋计算机博弈关键技术分析

徐心和　　王　骄

教育部暨辽宁省流程工业自动化重点实验室

东北大学人工智能与机器人研究所，沈阳，110004

(xuxinhe@ise.neu.edu.cn)

摘要：机器博弈被认为是人工智能领域最具挑战性的研究方向之一。国际象棋的计算机博弈已经有了很长的历史，并且经历了一场波澜壮阔的“搏杀”，“深蓝”计算机的胜利也给人类留下了难以忘怀的记忆。中国象棋计算机博弈的难度绝不亚于国际象棋，不仅涉足学者太少，而且参考资料不多。在国际象棋成熟技术的基础上，结合在中国象棋机器博弈方面的多年实践，总结出一套过程建模、状态表示、着法生成、棋局评估、博弈树搜索、开局库与残局库开发、系统测试与参数优化等核心技术要点，最后提出了当前研究的热点与方向。

关键词：人工智能，中国象棋计算机博弈，机器博弈过程建模，着法生成，评估函数，博弈树搜索

Key Technologies Analysis of Chinese Chess Computer Game

Xinhe Xu,　Jiao Wang

Institute of Artificial Intelligence and Robotics

Key Laboratory of Process Industry Automation, Ministry of Education

Northeastern University, Shenyang, 110004 China

(xuxinhe@ise.neu.edu.cn)

Abstract: Computer game is one of the most challenging topics in the field of artificial intelligence. Chess computer game has a long history, and come through tough research. Finally, DeepBlue’s victory startled the whole world. Chinese chess computer game is more complex than chess computer game, and the fewer researchers and fewer references lead the lag in the field. Based on a series of technologies of chess computer game and years practice of Chinese chess, a set of schemes and methods are proposed, such as the process modeling, state representation, move generation, evaluation function, searching game tree, opening book, endgame database, system test and parameter optimization, etc. Hot topics and tasks are also proposed at the end.

Keywords: Artificial Intelligence(AI), Chinese chess computer game, process modeling of computer game, move generation, evaluation function, searching game tree

1.　引言（Introduction）

博弈问题无所不在，小到孩童的游戏与争论、各种场合下的讨价还价，大到商家的竞争、各种突发事件（恐怖、灾害）的应急处理、国家的外交、流血的和不流血的战争，只要局中的双方主体存在某种利益冲突，博弈便成为矛盾表现和求解的一种方式。博弈与对策将成为一类智能系统研究的焦点问题。

[1]象棋是从两军对阵中抽象出来的一种智力游戏，因此它是博弈的一个标准问题。下棋的双方无时不在调动自己的一切智能，充分发挥逻辑思维、形象思维和灵感思维的能力。所以，在人工智能领域始终将棋类的机器博弈作为最具挑战性的研究方向之一。

早在半个多世纪之前，信息论的创始人C.E.香侬教授就提出了为象棋博弈编程的方案，成为机器博弈的创始人。[1] 半个世纪以来，国际象棋的计算机博弈十分活跃，而且确实经历了一场惊心动魄的激烈搏杀。[2]

[1]收稿日期：2005--

基金项目：国家自然科学基金资助项目（基金编号 60475036）

作者简介：徐心和 (1940 -)，男，黑龙江省哈尔滨人，东北大学人工智能与机器人研究所教授，博士生导师，研究方向与控制理论与应用，系统仿真，智能机器人，机器博弈等； 王骄 (1978 -)，男 ，辽宁省沈阳市人，东北大学博士研究生，研究方向为人工智能与机器博弈。

1958 年，IBM704 成为第一台能同人下棋的计算机，名为“思考”，思考速度每秒 200 步。到了 60 年代中期，科学家德里夫斯依然断言，计算机将无法击败一位年仅 10 岁的棋手。

1973 年，CHESS4.0 被 B. Slate and Atkin 开发出来，成为未来程序的基础。1979 年，国际象棋软件 4.9 达到专家级水平。

1981 年，CRAYBLITZ 新的超级计算机拥有特殊的集成电路，预言可以在 1995 年击败世界棋王。

1983 年，Ken Thompson 开发了国际象棋硬件 BELLE，达到了大师水平。

80 年代中期，美国的卡内基梅隆大学开始研究世界级的国际象棋计算机程序——“深思”。1987 年，“深思”首次以每秒钟 75 万步的思考速度露面，它的水平相当于拥有国际等级分为 2450 的棋手。1988 年，“深思”击败丹麦特级大师拉尔森。

1989 年，“深思”已经有 6 台信息处理器，每秒思考速度达 200 万步，但在与世界棋王卡斯帕罗夫进行的“人机大战”中，以 0 比 2 败北。

1993 年，“深思”二代击败了丹麦国家队，并在与世界优秀女棋手小波尔加的对抗中获胜。

1995 年，IBM“深蓝”更新程序，新的集成电路将其思考速度提高到每秒 300 万步。1996 年，“深蓝”在向卡斯帕罗夫的挑战赛中，以 2 比 4 败北。

1997 年，由 1 名国际特级大师，4 名电脑专家组成的“深蓝”小组研究开发出“更深的蓝”，它具有更加高级的“大脑”，通过安装在 RS／6000S 大型计算机上的 256 个专用处理芯片，可以在每秒钟计算 2 亿步，并且存储了百年来世界顶尖棋手的 10 亿套棋谱，最后“超级深蓝”以 3.5 比 2.5 击败了卡斯帕罗夫。卡斯帕罗夫要求重赛，但没有得到回应。

时至今日，尽管新的硬件、软件系统层出不穷，但国际象棋的机器博弈和人机大战仍然持续不断。因为人们总在不断地挑战自我，况且计算机在人机大战中仍然没有占据绝对优势。卡斯帕罗夫也仅仅输给“超级深蓝”那一次。

中国象棋是世界上历史最为悠久的棋类，早在 2000 多年前的战国时代就已经有了关于象棋的记载。然而中国象棋的计算机博弈却开展的不尽人意，成了“被爱情遗忘的角落”。缺少学者的关注，寥寥无几的参与者，匮乏的参考文献，沉寂的计算机博弈氛围，使得中国象棋的计算机博弈在中国大陆难有作为，只是成为一些商家的游戏软件和教学载体。这便是当前我们所面临的艰难局面。

国际象棋棋盘 8 行 8 列总计 64 格，中国象棋 10 行 9 列总计 90 个交点，显然中国象棋的运子空间更大。相比之下，中国象棋的着法更为特殊（如蹩马脚、压象眼等），棋局变化也更加复杂，这都是对中国象棋的计算机博弈提出的困难和挑战。

随着计算机博弈在Othello、Checker和国际象棋三种棋类上的成功，全世界的学者又把目光投到更为复杂的中国象棋(Chinese Chess)、日本将棋（Shogi）、围棋(Go)上面。几种棋类遍历搜索的复杂度对比见表 1-2，表中的数字为复杂度的自然对数值。显然，这更是对中国学者提出的严峻挑战。[3]

表 1-2　几种棋类的空间复杂度及树的复杂度对比

	空间复杂度	树的复杂度
Chess	50	123
Chinese chess	52	150
Shogi	71	226
Go	160	400

面对如此富于挑战性的局面，东北大学人工智能与机器人研究所从 2003 年便开始了中国象棋计算机博弈课题的研究。在认真借鉴国际象棋开发成果的基础上，突破了系列关键技术，开发出具有相当棋力的博弈软件系统，并在 2004 年成立了“棋天大圣”代表队，准备在中象机器博弈领域开创新的局面。本文便是针对系列关键技术的总结和归纳。接下来的各节分别论述了过程建模、状态表示、着法生成、棋局评估、博弈树搜索、开局库建立、残局库开发、系统测试与参数优化等核心技术要点，最后提出了当前研究的难点。

2. 象棋博弈过程建模（Modeling of chess computer game）

人工智能的先驱者们曾认真地表明：如果能够掌握下棋的本质，也许就掌握了人类智能行为的核心；那些能够存在于下棋活动中的重大原则，或许就存在于其它任何需要人类智能的活动中。

在各种棋牌博弈当中，象棋是一种完全知识博弈（Games of complete information），即参与双方在任何时候都完全清楚每一个棋子是否存在，位于何处。

象棋博弈具有如下基本特征：[1]

（1） 规则简单明确，成功与失败的判定标准简单，不包含任何机会或偶然性；

（2） 问题的状态(即局面)数量在数学意义上是有限的；

（3） 问题的解决在认识意义上不需要过多的知识。

为了深入探讨机器博弈的原理与方法学问题，尤其要从系统论和博弈论的角度研究问题的规律与求解方法，有必要分析象棋对弈的演化过程，建立相应的数学模型。图 2-1 给出了象棋博弈状态演化过程图。图中表明棋局状态是在着法算子作用下进行演化的，其对应的状态转移方程可以写成

$$S_{n+1} = S_n \cdot q_{n+1}, \quad S_0 = S(0) \qquad (2\text{-}1)$$

式中 S_0 为象棋的初始局面，q_{n+1} 为第 n+1 步的着法算子，而 S_{n+1} 为走完第 n+1 步后的棋局。于是，不难写出

$$S_F = S_0 \cdot q_1 \cdot q_2 \cdot \ldots \cdot q_F = S_0 \cdot Q \qquad (2\text{-}2)$$

式中 S_F 为终局，或红胜，或黑胜，或和棋。显然，着法序列 $Q = \{q_1 q_2 q_3 \ldots . q_F\}$ 便是记载博

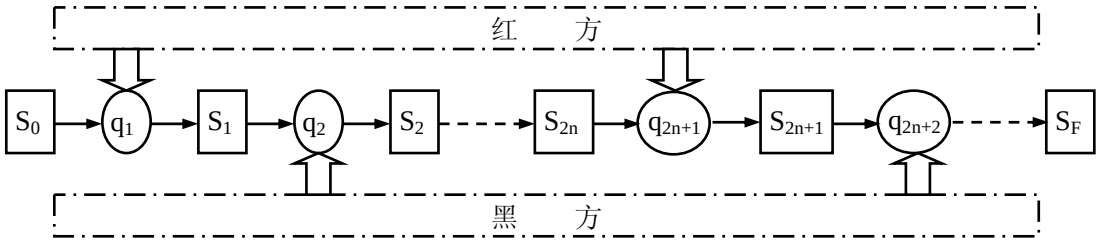

图 2-1　象棋博弈状态演化过程图

弈过程的棋谱，其中单数项序列 $Q_{odd} = \{q_1 q_3 \ldots .\}$ 为红方系列着法，而偶数项序列 $Q_{evn} = \{q_2 q_4 \ldots .\}$ 便是黑方的系列着法。

着法便是弈棋者的决策。尽管每次只能走出一步，但弈棋者可能考虑过全部有价值的走法，并且通过前瞻若干步，才形成弈棋者的当前决策。图 2-2 便以状态演化流程框图的形式展示了计算机应该如何实现这样的“思维”过程。

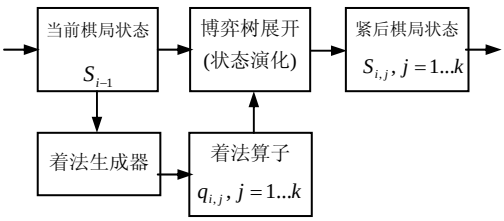

图 2-2　博弈者思维过程的机器实现流程

在这里全部可能的走法是通过着法生成器给出的，由它生成了第 i 步的 k 种可行着法，根据状态演化方程（2-1），在算子 $q_{i,j}, j=1...k$ 的作用下所形成的第 i 步后的棋局便不是一个，而同样是 k 个紧后棋局状态，即 $S_{i,j}, j=1...k$。显然，这便是决策树，亦称为博弈树的展开过程。弈棋者按照这样的思维方式，必然展开一颗规模庞大的、根在上而叶在下的博弈树，如图 2-3 所示。它与一般决策树的不同点则在于它的决策主体不是一方，而是相互对立的双方，即红方和黑方。图中方节点代表轮到红方走棋的状态（棋局），圆节点代表轮到黑方走棋的状态。显然，方、圆节点间的连线便对应于一个个着法。如果说，一个优秀棋手可以前瞻四步，则表示在他的脑海中的是将这棵博弈树展开 4 层（Depth）。

博弈树上的每一个节点都代表一个棋局，棋手就是要在众多的叶子节点上挑选一个最佳的局面作为自己的选择，从而反推到当前的着法。

中国象棋博弈树的庞大是可以想象的。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 $45^{90} \approx 10^{150}$ 种局面。据说这一天文数字要比地球上原子数目还要多，即使用世界上最快的计算机进行计算，直到地球毁灭也无法算出第一步的着法。

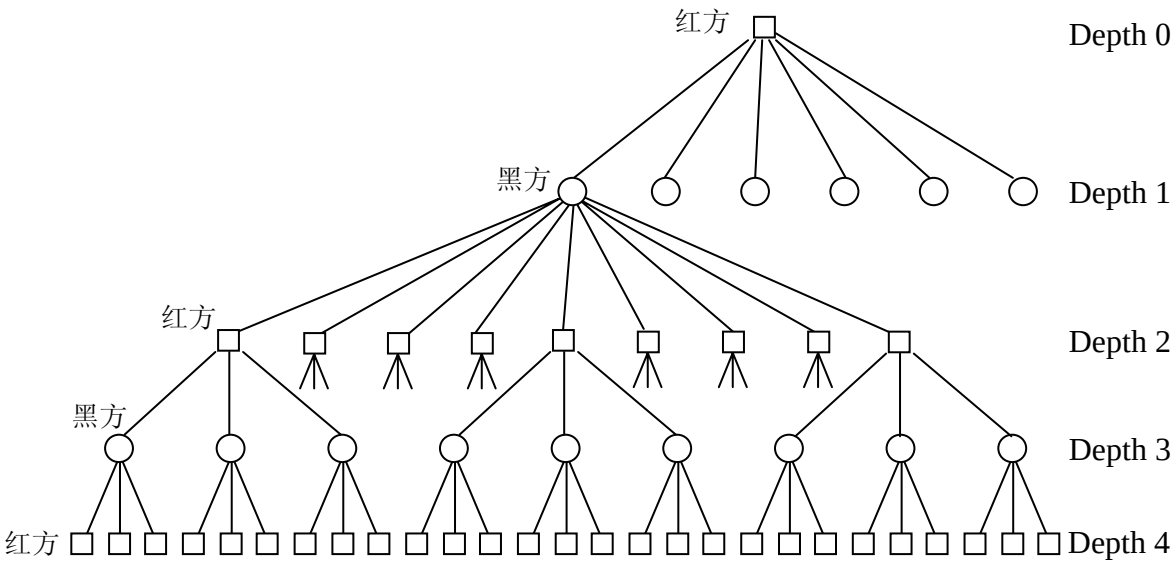

图 2-3 红方走棋时展开深度为 4 的博弈树

3. 棋局状态的表示（Board and state representation）

要让计算机下棋首先需要解决的就是棋盘和棋局的数字表示。棋盘上面有 9 路 10 行形成 90 个交叉点，它很容易用 10*9 的棋盘数偶矩阵 $M_{10\times 9}$ 表示与坐标的对应关系。

要表示棋局则首先要给棋子编码。应该说编码的方法是任意的，只要能够满足棋局表示的唯一性和可区分性，都是可行的编码。如果考虑和追求棋局处理的节俭与快捷，那么在编码上还是具有研究的余地。

“棋天大圣”的兵种（Arms）编码在表 3-1 中给出。红方和黑方兵种编码仅相差一个负号，这为棋局评估红黑双方保持对称性提供了方便。

仅有兵种的编码是不够的。为了跟踪棋子的挪动与拼杀，还需要进行棋子编码。我们的做法如表 3-2 所示。

表 3-1 象棋兵种的中英文表示与棋天大圣的兵种编码表

国象兵种	King	Rook	Knight	Cannon	Queen	Minister	Pawn
中象兵种	King	Rook	Horse	Cannon	Bishop	Elephant	Pawn
红子	帅	车	马	炮	仕	相	兵
字母代号	k	r	h	c	b	e	p
棋天大圣编码	1	2	3	4	5	6	7
黑子	将	车	马	炮	士	象	卒
字母代号	K	R	H	C	B	E	P
棋天大圣编码	−1	−2	−3	−4	−5	−6	−7

表 3-2 棋子（Chessman）编码（编号）表

j	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
黑方	将	黑车		黑马		黑炮		黑士		黑象		黑卒				
j	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32
红方	帅	红车		红马		红炮		红仕		红相		红兵				

由此便可以根据棋局给出兵种状态矩阵 S^B 和棋子状态矩阵 S^M 。式（3-1）（3-2）便给出了中象初始局面（图 3-1）对应的状态矩阵。

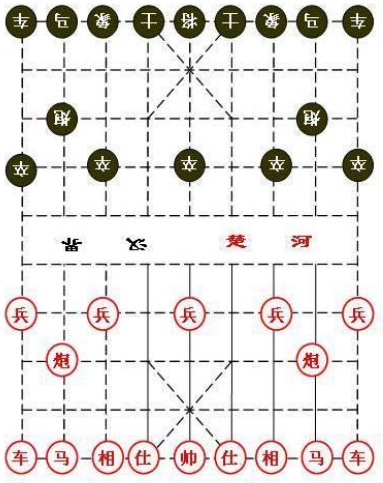

图 3-1 中象开局

$$S_0^B=\begin{bmatrix}-2&-3&-5&-6&-1&-6&-5&-3&-2\\0&0&0&0&0&0&0&0&0\\0&-4&0&0&0&0&0&-4&0\\-7&0&-7&0&-7&0&-7&0&-7\\0&0&0&0&0&0&0&0&0\\0&0&0&0&0&0&0&0&0\\7&0&7&0&7&0&7&0&7\\0&4&0&0&0&0&0&4&0\\0&0&0&0&0&0&0&0&0\\2&3&5&6&1&6&5&3&2\end{bmatrix} \quad (3\text{-}1)$$

$$S_0^M=\begin{bmatrix}2&4&10&8&1&9&11&5&3\\0&0&0&0&0&0&0&0&0\\0&6&0&0&0&0&0&7&0\\12&0&13&0&14&0&15&0&16\\0&0&0&0&0&0&0&0&0\\0&0&0&0&0&0&0&0&0\\28&0&29&0&30&0&31&0&32\\0&22&0&0&0&0&0&23&0\\0&0&0&0&0&0&0&0&0\\18&20&26&24&17&25&27&21&19\end{bmatrix} \quad (3\text{-}2)$$

为了避免从棋子状态矩阵中提取某棋子实时位置的搜索过程，还需要直接给出棋子位置矩阵：$P^M=\left[p_{i,j}^M\right]_{2\times32}$。矩阵中第一行 $p_{1,j}^M$ 表示编号为 j 的棋子在棋盘矩阵中的行号，矩阵中第 2 行 $p_{2,j}^M$ 表示编号为 j 的棋子在棋盘矩阵中的列（路）号。对应于 S_0^M 则有初始棋子位置矩阵为

$$P_0^M=\begin{bmatrix}1&1&1&1&1&3&3&1&1&1&1&4&4&4&4\\5&1&9&2&8&2&8&4&6&3&7&1&3&5&7&9\\10&10&10&10&10&8&8&10&10&10&10&7&7&7&7\\5&1&9&2&8&2&8&4&6&3&7&1&3&5&7&9\end{bmatrix} \quad (3\text{-}3)$$

在着法生成过程中，时常只关心棋子的分布，而不关心是什么棋子，这时可以用比特棋盘表示棋局的某种状态，它其实是棋盘状态矩阵 S 的布尔表示。比特棋盘的定义为

$$B=\left[b_{i,j}\right]_{10\times9}, b_{i,j}=\begin{cases}1 & s_{i,j}\neq0\\0 & s_{i,j}=0\end{cases} \quad (3\text{-}4)$$

根据计算过程的需要，还可以定义其它含义的各种单项比特矩阵或比特向量。如：红棋比特矩阵，中路比特向量等。通常我们使用状态集合 S_n 来表示 n 时刻的棋局状态。

$$S_n=\{S_n^B,S_n^M,P_n^M,B_n,...\} \quad (3\text{-}5)$$

4. 着法生成（Move generation）

棋手都知道中国象棋着法的表示方法，如：炮八平五，马 8 进 7 等等。看得出，这种着法是和当时的棋局有着不可分割的联系。我们称其为相对着法，用 q^R 来表示。而在机器博弈中所说的着法 q，都是绝对着法，它由提-动-落-吃 4 部分内容组成。即

$$q=\{from\quad moved\quad to\quad killed\} \quad (4\text{-}1)$$

式中 *from* 为“提址”，即此着的出发位置；*moved*（*chessman moved*）为“动子”，即走动哪个棋子；*to* 为“落址”，即着法的到达位置；*killed*（*chessman killed*）为“吃子”，即吃掉哪个棋子。显然，落址处有对方棋子，则形成吃子；落址处没有棋子，则“吃空”，仅走不吃；落址处为本方棋子，则为非法着法。

着法生成方法一般有棋盘扫描法、模板匹配法和预置表法，时常还结合使用。

4.1 棋盘扫描法（Scanning board method）

根据象棋规则，定义可行区域，如棋盘有效区域 $A=\{(i,j)|1\le i\le10,1\le j\le9\}$，红方半区 $A_R=\{(i,j)|6\le i\le10,1\le j\le9\}$，黑方九宫 $A_{PB}=\{(i,j)|1\le i\le3,4\le j\le6\}$ 等。在已知提址和动子的情况下，根据各兵种的行棋规则，计算合法落址。有时还要考虑制约条件。表 4-1 给出了马的着法规则。其它动子以此类推。

表 4-1 提址为（i,j）动子为马的着法计算规则

棋子 *moved*	落址 *to*	有效区域	制约条件 *Subject to*	吃子 *killed*
马	$(i\pm1, j+2)$	*A*	$(i, j+1)$ 无子	本方子则止对方子则吃
	$(i\pm1, j-2)$	*A*	$(i, j-1)$ 无子	
	$(i+2, j\pm1)$	*A*	$(i+1, j)$ 无子	
	$(i-2, j\pm1)$	*A*	$(i-1, j)$ 无子	

虽然在着法的表达上，棋盘扫描法最为直观，但在着法生成的过程中需要在棋盘上反复扫描有效区域、制约条件和落址状况，时间开销巨大，实战意义不强。

4.2 模板匹配法（Matching template method）

当动子确定之后，其落址与提址的相对关系便被固定下来。于是可以为某些动子设计“模板”，只要匹配到提址，便可以迅速找到落址。图 4-1 给出了走马的匹配模板。当将马字对准提址，×表示蹩马腿的制约条件，〇表示符合“走日”规则的落址，根据×的具体分布，很容易判断可能的落址。再通过单项比特矩阵比对，实现“本方子则止，对方子则吃”，完成“提-动-落-吃”内容的确定。

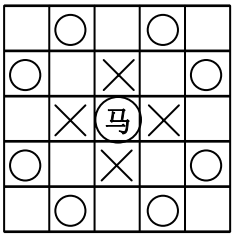

图 4-1 走马匹配模板

比较适合使用模板的动子为马和相（象）。

4.3 预置表法（Prefabricated table method）

预置表法是使用最为经常的着法生成方法。它的基本思想就是用空间换时间。为了节省博弈过程中的生成着法的扫描时间，将动子在棋盘任何位置（提址）、针对棋子的全部可能分布，事先给出可能的吃子走法和非吃子走法。

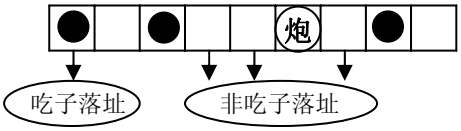

图 4-2 炮行着法预置表项举例

以图 4-2 炮的一个行着法预置表为例，炮的提址为某行的第 6 列，该行考虑的棋子分布为[101000010]布尔向量形式。不难得出，此时炮的非吃子着法的落址为[000110100]，而可能的吃子着法的落址为[100000000]。将这样的输入（炮的列号，该行棋子的布尔分布）和输出（吃子落址和非吃子落址的布尔表示）关系写入一个预置表中，在使用时通过查表，便很快可以得到可行的着法。而且还可以区分开吃子着法和非吃子着法，必然有利于搜索路径的选择（先吃，后不吃）。

这样的预置表的规模是比较大的。对炮而言（车也是这样），就每一行棋子的可行排列数恰好对应于 9 位二进制数（有子为 1，无子为 0），即 $2^9 = 512$ 种情况（项）。考虑到炮的 9 种可能位置，预置表的规模即为 9*512 项。再分开吃子和非吃子，还要考虑各列的全部情况，所以表的总规模为：2*9*512+2*10*1024=29696 项。如果假设每一项需要 4 个字节，那存放炮的着法预置表就需要 116k 内存空间。应该说还是可以接受的，因为它可以将着法生成的速度提高几个数量级。

5. 棋局评估函数（Chess evaluation function）

棋局评估就是给棋局打分。由于在较少的步数内，一般难以将对方将死。在难以判断输赢的情况下识别棋局的好坏，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机器博弈中最为人性化的部分。

目前对于评估函数取得共识的观点，应该包括如下部分：[4][5]

(1) 棋子价值评估值（e_1）。简单说就是评估双方都有哪些棋子在棋盘上。如设定：车-1000，马-450，炮-450，仕-170，相-160，兵-60，将帅-无穷大，红方为正值，黑方为负值。

(2) 棋子位置评估值（e_2）。同样一个棋子，处在棋盘的不同位置上其价值大不相同，式（5-1）给

$$e_{2(m=p)}=\begin{bmatrix}0&3&6&9&12&9&6&3&0\\18&36&56&80&120&80&56&36&18\\14&26&42&60&80&60&42&26&14\\10&20&30&34&40&34&30&20&10\\6&12&18&18&20&18&18&12&6\\2&0&8&0&8&0&8&0&2\\0&0&-2&0&4&0&-2&0&0\\0&0&0&0&0&0&0&0&0\\0&0&0&0&0&0&0&0&0\\0&0&0&0&0&0&0&0&0\end{bmatrix} \quad (5\text{-}1)$$

出了兵的位置评估值。可见过河兵，尤其是逼近和进入九宫的兵身价倍增。根据设计者的经验，各个兵种在其可行位置上都应该给出位置评估矩阵，以方便计算其位置评估值。而总的棋子位置评估值e_2则为全部盘上棋子评估值之和。

(3) 棋子灵活度评估值（e3）。对各兵种而言，每多一个可走位置就加上一定分值。如设定：兵-15，士-1，象-1，车-6，马-12，炮-6，将-0。

(4) 棋子配合评估值（e_4）。重点是车马炮的配合与牵制。过河兵牵手可加 200 分。连环马、担子炮、霸王车等都可以考虑加分。

(5) 将帅安全评估值（e_5）。此项多从盘势上加以考虑。如多子归边、空头炮、当头炮、沉底炮、将帅位置等，都要给出一定数量、有正有负的评估值。

我方总的棋局评估值便是求取各项评估值之和 $e^{red}=\sum_{i=1}^{k}e_i$。不仅要计算本方的，还要计算对方的。本方为正值，对方为负值，其代数和即为当前局面评估值。显然，总值为正对我方有利，负值对对方有利。绝对值的大小说明双方棋势的差距。

不难看出，评估函数中涉及到的权值系数可能多达上千个，都需要认真选择与权衡。困难之处在于，至今还有许多盲点，或者认识不到，或者说不出好坏，有待进一步研究与开发。可能还需要引进“自适应”与“自学习”功能。

6. 博弈树搜索（Searching game tree）

本文第 2 节已经为象棋博弈问题建立了状态空间显示图，亦即博弈树图模型（见图 2.3）。搜索法是求解此类图模型的基本方法。前面提到，我们无法搜索到最终的胜负状态，于是搜索的目标便成为如何在有限深度的博弈树中找到评估值最高而又不剧烈波动的最佳状态（棋局），于是从当前状态出发到达最佳状态的路径便为最佳路径（Principal continuation），它代表着理智双方精彩对弈的系列着法。而最佳路径上的第 1 步棋便成为当前局面的最佳着法（The best move）。所谓“不剧烈波动”就是说最佳棋局不是在进行子力交换与激烈拼杀的过程当中。

需要注意的是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。在图 2.3 中，红方为走棋方，它在偶数层的着法选择是要在其全部子节点中找到评估值最大的一个，即实行“Max搜索”。而其应对方——黑方在奇数层的着法选择则是在其全部子节点中要找到评估值最小的一个，即实行“Min搜索”。香侬(Claude Shannon)教授早在 1950 年首先提出了“极大-极小算法”（**Minimax Algorithm**）[1]，从而奠定了计算机博弈的理论基础。

在搜索策略上，一般可以分为以下三种[1][6]：

A 类——穷尽搜索（Exhaustive search）

B 类——选择性搜索（Selective search）

C 类——目标导向搜索（Goal oriented search）。

会下棋的人都知道“一着不慎，满盘皆输”的道理。于是，看得远（搜索的深），看得准（真正找到指定深度内的最佳的平稳棋局），便成为搜索算法的基本着眼点。

显然穷尽搜索成为人们首选的搜索策略。原始的蛮力搜索（Brute search）一般采用广度优先搜索(Breadth-first search)[6]，一层层展开，一层层搜索，因为“穷尽”而没有风险，不会漏掉展开深度内的最优解。但是，在给定的时间内搜索的节点数是基本固定的，当考虑到博弈树的平均分枝数（分枝因子B=40~50），搜索深度也是基本固定的。假设计算机搜索节点速率为 1M/秒，中国象棋B=45，表 6.1 给出了在不同的给定时间T内达到的搜索层数D。

表 6.1　给定时间可达搜索深度（1M 节点/秒）

T	1 秒	1 分	1 小时	1 天	1 年	10 年	100 年
D	3.62	4.70	5.78	6.62	8.17	8.77	9.38

由于完整的博弈树过于庞大，盲目搜索所能达到的层数十分有限，在象棋博弈中几乎没有实用价值。

若想在指定时间内将搜索深度加以提高，一方

面需要优化硬件和程序代码，提高单位时间内搜索的节点数；另一方面就需要像人类棋手一样有选择性地进行搜索，即对博弈树进行必要的裁剪（cut-off / pruning）。

6.1 α-β搜索（Alpha-Beta search）

α-β剪枝搜索是一种基于**α-β**剪枝（**α-β** cut-off）[8]的深度优先搜索（Depth-first search）。为了表述方便，我们不妨将走棋方定为MAX方，因为它选择着法时总是对其子节点的评估值取极大值，即选择对自己最为有利的着法；而将应对方定为MIN方，因为它走棋时需要对其子节点的评估值取极小值，即选择对走棋方最为不利的、最有钳制作用的着法。

在对博弈树采取深度优先的搜索策略时，从左路分枝的叶节点倒推得到某一层MAX节点的值，可表示到此为止得以“落实”的着法最佳值，记为**α**。显然此**α**值可作为MAX方着法指标的下界。在搜索此MAX节点的其它子节点，即探讨另一着法时，如果发现一个回合（2 步棋）之后评估值变差，即孙节点评估值低于下界**α**值，则便可以剪掉此枝（以该子节点为根的子树），即不再考虑此“软着”的延伸。此类剪枝称为**α**剪枝。图 6.1 给出了搜索和剪枝过程，最后得到如粗箭头所示的最佳路径片断。

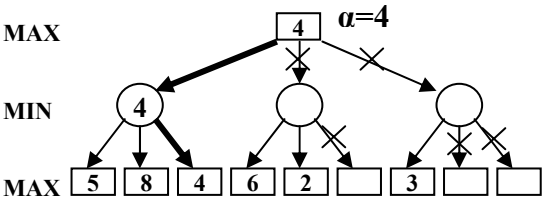

图 6.1　**α**剪枝示意图

同理，由左路分枝的叶节点倒推得到某一层MIN节点的值，可表示到此为止对方着法的钳制值，记为**β**。显然此**β**值可作为MAX方可能实现着法指标的上界。在搜索该MIN节点的其它子节点，即探讨另外着法时，如果发现一个回合之后钳制局面减弱，即孙节点评估值高于上界**β**值，则便可以剪掉此枝，即不再考虑此“软着”的延伸。此类剪枝称为**β**剪枝。图 6.2 给出了搜索和剪枝过程，最后得到如粗箭头所示的最佳路径片断。

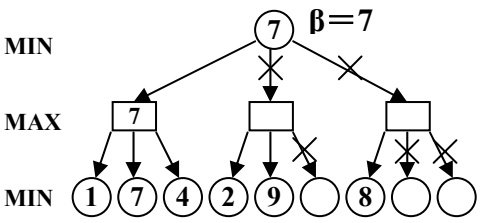

图 6.2　**β**剪枝示意图

需要指出的是，**α-β**剪枝是根据极大-极小搜索规则的进行的，虽然它没有遍历某些子树的大量节点，但它仍不失为穷尽搜索的本性。剪枝技巧的发现，一下便为博弈树搜索效率的提高开创了崭新的局面。[8]

1975年Knuth和Moore给出了**α-β**算法正确性的数学证明。[9] **α-β**剪枝算法的效率与子节点扩展的先后顺序相关。在最理想情况下（极小树），其生成的节点数目为

$$N_D = 2B^{D/2} - 1 \qquad \text{（D为偶数）} \quad (6\text{-}1)$$

$$N_D = B^{(D+1)/2} + B^{(D-1)/2} - 1 \text{(D 为奇数)} \quad (6\text{-}2)$$

其中D为搜索深度，B为分枝因子（Branching factor）。在不使用**α-β**剪枝时，需要搜索的节点数是 $N_D = B^D$，即极大树。所以最理想情况下**α-β**算法搜索深度为D 的节点数相当于搜索深度为D/2 的不做任何剪枝节点数。

如何才能得到极小树？不难看出，如果最左路的分枝就是最佳路径，亦即理智双方最为精彩的对弈着法序列，那么就可以将右路各分枝陆续剪掉，从而使搜索的节点数仅为极大树的 4.4%（2/B）。

为了得到最好的节点扩展顺序，许多搜索算法在着法（节点扩展的分枝）排序上给予特别的关注。比如在着法生成（节点扩展）时，先生成吃子着法，尤其先生成吃分值高的“大子”着法，因为由此产生着法更有可能是最佳的。[9]

围绕着法排序，已经出现许多优秀的搜索算法与举措。如：同形表法（Transposition table）[10]、吃子走法的SEE排序、杀手走法（Killer heuristic）[11]、未吃子走法的历史启发排序（Historic heuristic）[12]、类比法（Method of analogies）[13]等。

有人将**α-β**剪枝作用延伸到多个回合之后，于是又出现了深层**α-β**剪枝（Deep **α-β** cut-off）算法。[14]也取得很好效果。

6.2 α-β窗口搜索（α-β windows search）

从**α-β**剪枝原理中得知，**α**值可作为MAX方可实现着法指标的下界，而**β**值 （应对方的钳制值）便成为MAX方可实现着法指标的上界，于是由**α**和**β**可以形成一个MAX方候选着法的窗口。围绕如何能够快速得到一个尽可能小而又尽可能准确的窗

口，也便出现了各种各样的**α-β**窗口搜索算法。如Fail-Soft Alpha-Beta[9]、Aspiration Search（渴望搜索）[15]、Minimal Window Search（最小窗口搜索）[16]、Principal Variable Search（PVS搜索）/ Negascout搜索[16]、宽容搜寻（Tolerance Search）[16]等。

6.3 迭代深化搜索（Iterative deepening search）

不难想像，深度为D-1 层的最佳路径，最有可能成为深度为D层博弈树的最佳路径。Knuth和Moore分析表明[9]，对于分枝因子为B博弈树，利用**α-β**剪枝搜索D层所需时间大约是搜索D-1 层所需时间的$\sqrt{B}$ 倍。如果国象取B=36，每多搜一层就要花上原先的 6 倍时间。于是CHESS4.6 和DUCHENSS课题组开始采用逐层加深搜索算法[17]。先花 1/6 的时间做D-1 层的搜索，找到最佳路径，同时记载同形表、历史启发表、杀手表等有价值信息，以求达到D层最好的剪枝效果。可谓“磨刀不误砍柴功”。

目前几乎所有高水平的博弈程序都采用迭代深化算法，并在不断改进。如PV记录(Principal Variation) [10]，以及和渴望窗口搜索（Aspiration Windows Search）[15]的结合，都会对走法排序产生非常好的效果。另外，逐层加深的搜索算法比固定深度搜索算法更适合于对弈过程的时间控制。

6.4 启发式搜索（Heuristic search）

具体问题的领域决定了初始状态、算符和目标状态，进而决定了搜索空间。因此，具体问题领域的信息常常可以用来简化搜索。此种信息叫做启发信息，而把利用启发信息的搜索方法叫做启发性搜索方法。[6]

利用象棋的领域知识（启发信息）设计博弈搜索的启发式算法或方法，在着法排序中就有许多成功的应用。这里需要特别提及的则是平静搜索、空着搜索和兑子搜索等。

平静搜索（Quiescent Search）[18]。**α-β**搜索使用的是给定深度搜索，当局面变化剧烈的时候，虽然已经达到搜索的最大深度，但此时评估函数的返回值并不能准确地表示当前局面的情况。举个简单的例子，某个叶节点上红车吃掉了黑炮，但是下一步红车会被黑方吃掉。如果在此叶节点调用评估函数，返回值肯定对红方十分有利，但会引起判断失误。平静搜索判断局面是否剧烈振荡的准则是走棋方是否还有吃子走法，一直延伸到走棋方无吃子走法，也就是相对平静的局面。

将军延伸（Check Extension）[19]，是指当本方受到对方将军时所进行的扩展。由于逃避对方对本帅攻击的解将着法不多，所以我们可以对当前节点的搜索深度增加一层，以更准确地评估攻击的危险性。

唯一着法延伸（One_Reply Extension）[19]。当本方受对方将军的时候，并且解将着法只有一步，这时候由于搜索量不大，我们在将军延伸之外，还要对其进行额外的延伸。

兑子延伸（Recapture Extension）[19]。搜索过程中，如果出现A棋子吃掉对方的B棋子，随即A棋子又被对方吃掉，那么也要对这样的局面进行延伸，以保证对兑子进行准确的评估。

空着搜索（NullMove）是在1993年由Chrilly Donninger最先提出的[20]。NullMove的思想是放弃本方的走棋权利，让对方连续走棋，如果得到的分值还大于原先的**β**值，说明对方没有“硬着”可施，于是在此分枝下搜索的意义已经不大，免于搜索。NullMove危险性比较小，实现较为简单，剪枝效果明显，现已被棋类博弈广泛采用。

6.5 负极大值算法（NegaMax algorithm）

前面谈到博弈树的搜索是一种“变性”搜索。在偶数层进行“Max 搜索”，而在奇数层进行“Min搜索”。这无疑给算法的实现带来一大堆麻烦。

Knuth和Moore充分利用了“变性”搜索的内在规律，在 1975 年提出了意义重大的负极大值算法[9]。它的思想是：父节点的值是各子节点值的变号极大值，从而避免奇数层取极小而偶数层取极大的尴尬局面。

$$F(v)=\max\left\{-F(v_1),-F(v_2),\cdots-F(v_n)\right\} \quad (6\text{-}3)$$

其中v_1，v_2……v_n为v的子节点。

此时需要特别注意的则是，如果叶节点是红方（走棋方）走棋，评估函数返回*RedValue-BlackValue*，如果是黑方（应对方）走棋，则返回*BlackValue -RedValue*。另外，由于负极大值计算等价于“Min搜索”，所以这里只进行**β**剪枝，非常有利于编程实现和提高搜索速度。

从以上有限的介绍不难看出，博弈树的搜索算法丰富多彩，改革、重组与创新的余地很大，一定会成为“兵家必争之地”，成为机器博弈研究的重点。

7. 开局库与残局库（Opening book and endgame database）

7.1 开局库设计（Opening book）

象棋博弈一般分为三大阶段：开局、中局、残局。虽然有时在中局就已经决出了胜负而没有残局，但所有的对局都必须有开局，只不过长短不同。

中国象棋的开局是指棋局从初始状态开始的10～20个回合之内，对战双方各自展开子力，占据棋盘的有利位置。中国象棋讲究的是快速出动子力，各棋子协调作战，并且尽早占据中心位置。如果让搜索引擎从棋局一开始就进行搜索，那么搜索引擎能够看到的至多十几层之内的变化，很容易在战略上犯错误。因为电脑往往为了局部很小的利益而忽视全局的发展。中国象棋棋谱上记载的是经过千锤百炼的开局着法，这些着法公认是对全局的发展大有裨益的。如果把一些公认为最佳的开局存储在计算机中，在开局时用查询取代搜索和评估，那么会大大提高计算机在开局的对弈水平。达到这个目标的途径就是开发和利用开局库。

开局库中存放了数以十万计甚至更多的棋局，如何能够快速准确地搜索到当前对应的棋局成为开局库设计的关键技术之一。本文第3节给出的棋局状态描述方式显然无法适应此项需要。国际象棋的成功经验表明，开局库最好是采用Zobrist哈希技术加以实现[21]。

将棋局转换为哈希数，即哈希技术的基本原理是将盘上双方棋子的兵种代码（见表3-1）和坐标位置（3-3式）作为64位随机数发生器（*Random64*）的输入变量，将盘上全部棋子的对应的随机数异或求和，这个64位的哈希数和便作为给定棋局的索引值，用作棋局的存储与查询。哈希数的最大优点就在于它求的是64位数的异或和，当在着法（提-动-落-吃）的作用下，棋局发生了变化，只要将相应变化棋子的哈希数再异或一次，便可以转变成新局面对应的哈希数。亦即存在与（2-1）式相似的哈希值转移方程

$$H_{n+1} = H_n \cdot q_{n+1}, \quad H_0 = H(0) \qquad (7\text{-}1)$$

式中

$$H_n = \bigoplus_{j=1}^{32} Random64(m_j, (P_n^M)_j) \ ,$$

（$\oplus$为异或算子） （7-2）

开局库中与棋局索引值相联系的便是可以采用的着法，常常不止一个，每个着法都对应有输赢统计比率、作者偏好权重等，以便使用时选择。一般还应该具有自学习的功能，将新的对弈结果补充进来，并且不断自行调整比率与权重值。

7.2 残局库设计（Endgame database）

残局是指由少数残余子力所构成并进入决定胜负阶段的对峙局面。残局阶段的任务是，如何巩固和扩大既得的优势而赢得对手，或者在已处于劣势的情况下如何力争求和。在残局阶段，象棋大师的知识、人类的直觉与灵感往往能够战胜程序的搜索能力。因此，一般计算机在残局阶段是不占优势的。

提高残局阶段的棋力，比较常用的是残局库技术。在国际象棋中通常使用回溯法（Retrograde algorithm）构造残局库。把成熟的残局着法信息事先存储在残局局面的数据库文件中，当进入残局时只要存在与该局面相同的残局数据库文件，就可以从残局库中直接提取处理，这时不仅节省大量的评估和搜索时间，而且其走法策略都是完美的。

国际象棋残局库有3种[22]：赢/平局/输(win/draw/loss)残局库，将杀步数(distance-to-mate)残局库，变换步数(distance-to-conversion)残局库。各有特长，尚无定论。

鉴于残局库信息量庞大，对残局数据库的索引方案和存储压缩，都是不容忽视的任务[23]。另外，残局阶段对于象棋规则的考虑应该给与特别的关注。比如对于“长将”的评判，还有60步规则[24][25]（60-move-rule）等，如果给与忽略，在实际的对战中就可能判负。

8. 系统开发、测试与参数优化（System development, test and parameter optimization）

8.1 系统开发与实现（System development and realization）

目前比较有水平的象棋博弈软件的程序规模都比较庞大，源程序少则几万条，多则数十万条。从头开发是软件工程领域的艰巨任务。只有认真掌握了全部算法，认真做好总体设计和详细设计，才有成功的希望。

幸好，目前已经出现一些国象和中象的开源软件。[26][27][28]借助已有的人机界面和搜索引擎不断消化、改造、完善和丰富，从而形成具有自己特色和知识产权的博弈软件，应该说是最为现实的开发方案。

这里对于系统开发需要格外提出的几个问题是：

(1) 本文为了将博弈原理表述清楚，几乎所有的表达式都是采用矩阵形式，亦即二维数组。在程序运行过程中，二维数组的计算要比一维数组更多地耗时，因此都要转为一维数组进行编程。

(2) 由于博弈树展开的规模十分庞大，一般中局搜索的节点数可达千万个，因此节点信息的存储内容与方式便要非常讲究，否则内存空间危机将成为程序运行的薄弱环节。

(3) 相对运行空间而言，恐怕运行时间的矛盾更为突出，它是影响搜索深度和质量的瓶颈。如何以空间换时间，是各种算法研究的焦点问题。本文在着法生成一节中介绍的预置表法，便是解决此类问题的典型范例。

(4) 探讨人类博弈的认知过程，结合象棋对弈过程中出现的实际问题，研究新型的启发式搜索算法，将是机器博弈获得升华的不竭动力。

(5) 就国内而言，中国象棋残局库的开发基本还是空白。

8.2 系统测试与参数优化（System test and parameter optimization）

系统测试的重要性是不言而喻的。系统测试除了要检查系统需求定义的功能、排除编程中的错误、保证系统在对弈过程中顺利运行之外，更重要的是参数的优化与棋力的提高。前面提到在评估函数中就有成百上千个参数，在搜索过程中也有一系列的预置参数，都需要在调试和实际对弈过程中不断改进。

测试的最好办法是通过对战平台，可以是自己对自己，也可以是与其它象棋软件自动对阵。每当程序作了较大的修改，都需要通过对战检查会否发生死机或出现新的问题。

为了做到参数优化需要给出相当数量的测试棋局。由于事先知道棋局正解（最佳着法），检查被测软件是否找到了正解，研究为什么出现了偏离。一般出现偏离原因或是在评估中缺少了什么，或是权值给的不好。软件调试的办法，就是设法调出正确结果。然而常常是“按下去葫芦又起来了瓢”，因此参数优化是一个非常需要耐心、又非常需要象棋专业知识的细活，聘请象棋高手和象棋大师参加工作是必不可少的。

9. 结语（Conclusion）

总体来说，中国象棋的机器博弈还处在起步阶段，许多关键技术的研究还不够成熟，许多国际象棋成熟的做法我们还需要结合中国象棋的特点加以完善。不过，挑战中国象棋大师、特级大师和冠军的时刻指日可待。

需要特别指出的是，真正应用系统论、控制论和博弈论的方法来探讨象棋博弈问题就是在国际上也不多见。如何通过形式化和模型化来提升象棋博弈问题的研究，如何将象棋机器博弈的研究成果应用到社会安全与军事博弈当中，都是既有理论意义又有应用前景的科研项目。离散事件动态系统、模糊逻辑、神经网络、模式识别、参数估计、离散优化、数据挖掘等在机器博弈这一崭新领域大有用武之地。

参考文献：

[1] Shannon, Claude E., Programming a computer for playing chess[J], Philosophical Magazine, Vol. 41:256-275, 1950.

[2] http://www.chessit.net/file_topic/computerchess/c_briefhistory.htm (国际象棋人机博弈简史)

[3] Allis, L.V.. Searching for Solutions in Games and Artificial Intelligence[D]. Ph.D. Thesis, University of Limburg, Maastricht, The Netherlands. ISBN 90-9007488-0. 1994

[4] 王小春，PC 游戏编程[M],重庆：重庆大学出版社, 2002
(Wang xiao-chun, PC Game programming, ChongQing University press 2002:1-27)

[5] 胡达，C++Builder 程序设计范例——中国象棋[M]，清华大学出版社, 2002
(Hu Da, C++Builder Programming Paradigm－ChineseChess,Tsing Hua Press, 2002)

[6] 蔡自兴，徐光佑，人工智能及其应用[M]，清华大学出版社，2003.9
(Cai Zhi-xing, Xu Guang-you, Artificial Intelligence Principles and Applications, Tsing hua Press, 2003-9)

[7] A. Newell, J.Shaw amd h.Simon, Chess Playing Programs and the Pruning[J], IBM Journal,Oct.,1958, pp320-335

[8] 许舜钦，电脑对局的搜索技巧[J]，国立台湾大学工学学刊，第 51 期（1991.2），pp17－31
(Shun-Chin Hsu, Search Techniques for

Computer Game Playing, Taiwan University Press, Vol51(1991.2), pp17－31)

[9] D.E Knuth and R.N Moore，An analyze of Alpha-Beta Pruning[J], Artificial Intelligence, Vol,6,1975, pp.293-326

[10] David J.Slate and Laweence R.Atkin, Chess4.5 – The Northwestern University Chess Program, Chess Skill in man and machine[J], New York, Springer-Verlag,1997, pp82-118

[11] Selim G, Akl and Monroe M. Newborn, The Principal Continuation and the Killer Heuristic[C], Proceeding of ACM National Conference, Seattle, 1997, pp466-478

[12] Jonathan Schaeffer, The History Heuristic and Alpha-Beta Search Enhancements in Practice, IEEE Transactions on Pattern Analysis and Machine Intelligence. 1989. Vol.11. pp1203 - 1212.

[13] G,M. Adelson-Velesky, V.L, Arlazarov and M.V. Donsky, Some methods of Controlling the Tree Search in Chess Programs[J], Artificial Intelligence, Vol.6,1975, pp361-371.

[14] S.H. Fuller, J.G. Gasching and J.J.Gillogly, An Analysis of the Alpha-Beta Pruning Algorithm[D], Department of Computer Science Reportm Caregie-Mellon Univeristy, Pittsburg, 1973.

[15] Hermann Kaindl, Reza Shams, and Helmut Horacek, Minimax Search Algorithms with and without Aspiration Windows[J], IEEE transactions Pattern Analysis and Machine Intelligence, Vol 13, NO 12, December 1991, pp1225-1235

[16] Owner's Handbook, Chess Program Design[M], Chap.6, Turbo Pascal GameWorks, Ver.4.0, Borland, 1987

[17] M. Newborn, Recent Progress in Computer Chess[J], Advances in Computer, Vol.18.1978, pp59-117.

[18] Don F. Beal, A Generalised Quiescence Search Algorithm, Department if Computer Science, Queen Mary College[J], London University, Artificial Intelligence 43(1990), pp85－98

[19] Ed Schroder, How Rebel Plays Chess, http://members.home.nl/matador/chess840.htm

[20] Chrilly Donninger. Null Move and Deep Search: Selective-Search Heuristics for Obtuse Chess Programs[J], ICCA Journal, vol. 16, no.3, pp. 137-143, 1993

[21] Zobrist A. A New Hashing Method with Application for Game Playing[R]. Technical Report88, Computer Science Department, University of Wisconsin, Madison. 1970.

[22] Wu R, Beal D F. A Memory Efficient Retrograde Algorithm and Its Application to Chinese Chess Endgames [J]. More Games of No Chance MSRI Publications Volume 42, 2002, 207-228

[23] Yen S J Chen J C, Yang T N. Computer Chinese Chess[J]. ICGA Journal, Vol. 28, No.3, September 2005, pp 182-184.

[24] Chen S H. Design and Implementation of a Practical Endgame Database for Chinese Chess[D]. M.Sc. Thesis, Department of Computer Science and Information Engineering, National Taiwan University, Taiwan, 1998.

[25] Fang H R, Hsu T S, Hsu S C. Indefinite Sequence of Moves in Chinese Chess Endgames[C]. Proceedings of the Third International Conference on Computers and Games, 2002, pp 264-279

[26] ftp://ftp.cis.uab.edu/pub/hyatt/ (国际象棋象棋引擎crafty网站)

[27] http://www.wbec-ridderkerk.nl/ (国际象棋象棋引擎Fruit网站)

[28] http://www.elephantbase.net (中国象棋引擎象眼网站)

注：本文正式发表在《小型微型计算机系统》第 27 卷第 6 期（Vol.27, No.6, 2006）pp961-969