智能五子棋算法的设计实现

王长飞，蔡 强，李海生
（北京工商大学计算机学院，北京 100048）

摘 要：博弈是人工智能的主要研究领域之一。以五子棋为例，探讨**人机博弈中推理技术、搜索方法**和**决策规划**的实现算法。在 Visual C++环境下设计了一个基于上述算法的智能的五子棋程序，实现人机博弈。最后探讨了**优化五子棋算法的思路**，并给出算法实例。相比已有程序，实例在对弈水平和搜索效率方面均有显著的提高。

关键词：五子棋；估值；Alpha-Beta 搜索；优化
中图分类号：TP301.6 **文献标识码**：A **文章编号**：1004-731X (2009) 04-1051-04

Design and Implementation of Intelligent Gobang Playgame

WANG Chang-fei, CAI Qiang, LI Hai-sheng
(School of Computer Science, Beijing Technology and Business University, Beijing 100048, China)

Abstract: Gambling and chess is one of major research area in artificial intelligence. Based on gobang playgame, the *human-computer game reasoning, search methods* and *decision-making planning algorithm* were discussed. Then, an intelligent gobang system was designed and realized in Visual C ++ according to the mentioned algorithms. Finally, *some measures to optimize the gobang algorithm* were proposed and an application example was given. The application shows prominent improvement on both the playing level and search efficiency comparing to the existing programs.
Key words: gobang; evaluation; alpha-beta search algorithm; optimization

引 言

人工智能是一门综合性很强的边缘科学，它研究如何使计算机去做那些过去只能靠人的智力才能做的工作。而博弈是人工智能研究的一个重要分支，它不仅存在于游戏、下棋之中，也存在于政治、经济、军事和生物竞争中。

五子棋是起源于中国古代的传统黑白棋种之一。现代五子棋日文称之为“连珠”，英译为“Renju”，英文称之为“Gobang”或“FIR”（Five in a Row 的缩写），亦有“连五子”、“五子连”、“串珠”、“五目”、“五目碰”等多种称谓[1]。

与其他棋类相比，五子棋每一层搜索节点数量庞大，规则简单，估值函数可以做到比较细致。本文以五子棋为例研究人机博弈，在传统 Alpha-Beta 搜索结合估值函数方法的基础上，提出一系列优化搜索的措施，设计完成一个智能程度较高的五子棋系统。

1 传统五子棋算法介绍及初步实现

一般地，传统五子棋的算法主要包括：估值函数、搜索算法和胜负判断等。

收稿日期：2007-08-10 **修回日期**：2008-05-12
基金项目：北京市自然科学基金（4062010）
作者简介：**王长飞**(1985-)，男，江西赣州人，瑶族，硕士生，研究方向为计算机图形学；**通讯作者蔡强**(1969-)，男，重庆永川人，博士，副教授，中国系统仿真学会会员，研究方向为计算机图形学，科学计算可视化，数字娱乐技术；**李海生**(1974-)，男，山东宁津人，博士，副教授，研究方向为计算机图形学，科学计算可视化。

1.1 估值函数

不同的棋型，其优先级不同。例如，四个棋子连成一线且还能继续落子的棋型（活四）显然要比只有三个棋子连成一线（活三或死三）好。要使计算机正确地做出这种判断，就要把第一种棋型的估值设高。事实上，对于每一种特定的棋型，都需要相应的估值来反映其优劣情况。另外，由于搜索模块频繁地调用估值函数，为了尽可能地加快搜索速度，估值函数应设计的越仔细越好。

估值时，需要从四个方向上来考虑所下棋子对当前盘面的影响。这四个方向分别是以该棋子为出发点，水平、竖直和两条为 45 度角和 135 度角的线。

为方便分析棋盘上的格局，本文中约定以“A”代表黑子，“B”代表白子，“？”代表棋盘上空位。算法中关于棋子死活的规定如下：一方落子后，它的落子连成的一条线有两条不损伤的出路，则称该棋型是活的。否则称该棋型是死的。比如关于活三的定义：不论对手如何落子，仍然至少有一种方法可以冲四。因此，B?AAA? B 中的三个 A，不能算是活三；B?AAA??B 中的三个 A，也不是活三，尽管它有可能成为活四。这样，棋型的估值设计才能比较细致。本文算法对特定棋型的估值如表 1 所示。

1.2 Alpha–Beta 搜索

在博弈问题中，每一个格局可供选择的行动方案都有很多，因此会生成十分庞大的博弈树[2]。一般地只生成一定深度的博弈树，然后进行极小极大搜索。极大极小搜索是指：

表 1 特定棋型的估值

棋型名称	棋型模式	估值
活四	?AAAA?	300000
死四 A	AAAA?	2500
死四 B	AAA?A	3000
死四 C	AA?AA	2600
活三	??AAA??	3000
死三 A	AAA??	500
死三 B	?A?AA?	800
死三 C	A??AA	600
死三 D	A?A?A	550
活二	???AA???	650
死二 A	AA???	150
死二 B	??A?A??	250
死二 C	?A??A?	200

在一棵博弈树中，当轮到甲走时，甲定会选择子节点值最大的走法；而轮到乙走时，乙则会选择子节点值最小的走法[3]。使用估值函数对博弈树的每一个局面进行估值后，就可以通过极大极小搜索在博弈树中寻找最佳的合法走法。

在极大极小搜索的过程中，存在着一定程度的数据冗余。如图 1 左半部所示的一棵极大极小树的片断。其中节点下方数字为该节点的值，方形框节点代表计算机走，圆形框节点代表人走。A 节点表示计算机走，由于 A 是极大值点，根据极小极大搜索原理它要从 B 和 C 当中选最大的值。假设目前已经通过估值得出 B 为 18，当搜索 C 节点时，因为 C 是该人走，所以根据极小极大搜索原理要从 D、E、F 中选取最小的值。此时如果估出 D 为 16，那么 C 的值必小于或等于 16。又因为已经得出 B 的值为 18，说明节点 A 的值为 Max（B，C）=18，也就是说无须求出节点 C 的其他子节点如 E、F 的值就可以得出父节点 A 的值。这种将节点 D 的后继兄弟节点剪去的方法称为 Alpha 剪枝。同理，在图 1 右半部一棵极大极小树的片段中，将节点 D 的后继兄弟节点剪去称为 Beta 剪枝。

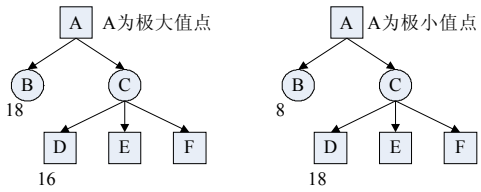


图 1 Alpha-Beta 剪枝

将 Alpha 剪枝和 Beta 剪枝加入极大极小搜索，就得到 Alpha-Beta 搜索算法，该算法描述如下：

```
int AlphaBeta(int depth, int alpha, int beta)
{
  if depth 为 0，说明当前节点是叶子节点 then
      返回对当前棋局的估值
  else
while 还存在可能的走法
{
走一步棋，从对手角度进行-AlphaBeta(depth-1,-beta，-alpha)的递归搜索，记录返回值为 val
撤消刚才走的一步
若 val 大于等于 beta，则返回 beta 的值
若 val 大于 alpha，则修改 alpha 的值为 val
}
    end while
end if
  返回 alpha
}
```

其中 depth 记录搜索的深度，alpha 记录搜索到的最好的值，beta 记录对于对手来说最坏的值。如果 INFINITY 表示无穷大，则 AlphaBeta(3, -INFINITY, INFINITY)表示完成一次 3 层的搜索。

1.3 胜负判断

在棋局的胜负是根据最后一个落子的情况来判断的。此时需要查看四个方向，即以该棋子为出发点的水平，竖直和两条分别为 45 度角和 135 度角的线，看在这四个方向上的其它棋子是否能和最后落子构成连续五个棋子，如果能的话，则表示这盘棋局已经分出胜负。

实际上，我们可以提前若干步预判当前棋局的胜负情况。本文算法采用了如下的规则对胜负进行预判，提高了算法的智能。在甲和乙对弈的棋局中，某个时刻轮到甲下棋时几种可能获胜的情况：

甲已有任意组活四，或者甲已有任意组死四：一步获胜

甲已有任意组活三，或者甲已有多于一组的死三：两步获胜

甲已有一组死三和任意组的活二：三步获胜

1.4 数据结构

本文以 15×15 的五子棋棋盘为例，实现算法。在算法中采用的数据结构包括：

以数组形式保存当前棋盘的盘面情况：

int m_nBoard[15][15]//当前棋盘，数组的每一个元素对应棋盘上的一个交叉点，用‘0’表示空位、‘1’代表白子、‘2’代表黑子。

int Computer[15][15][4],Player[15][15][4]//用来存放棋型数据。棋盘上每个点都可以在水平、竖直、45 度角线和 135 度角的线 4 个方向构成不同的棋型，所以一个点有 4 个记录。本文算法中 0 代表水平方向，1 代表竖直方向，2 代表 45 度角方向，3 代表 135 度角方向。例如，根据本文估值函数的约定，Computer[4][7][0]=2500 表示若计算机在（4，7）落子，将得到一个水平方向的死四。

m_wXCur1，m_wYCur1，m_wXCur2，m_wYCur2//记录黑白双方上一落子位置，以实现悔棋的功能。

2 五子棋算法的优化

到目前为止，我们使用传统的 Alpha-Beta 搜索结合估值函数的五子棋算法完成一个简单的五子棋对弈程序。虽然估值尽力做到细致、全面，但由于 Alpha-Beta 搜索存在博弈树算法中普遍存在的一个缺点—随着搜索层数的增加，算法的效率大大下降。所以搜索的效率还是不理想，五子棋程序的“智力”也不高。

因此，在上述基础上，我们继续研究，通过对 Alpha-Beta 搜索算法的优化与修正，针对五子棋本身的特点和规律，提出采取以下优化措施，显著地提高了五子棋程序对弈的水平和能力。

2.1 减少搜索范围

五子棋棋盘大小为 15×15，传统算法中计算机每走一步都要遍历整个棋盘，对于棋面上所有空位都进行试探性下子并估值，这样大大影响算法的效率。其实在某个时刻，棋盘上很多的位置都是可以不用去考虑的。如图 2 所示，根据经验，只要考虑距以棋子为中心边长为 4 的正方形区域即可。这样便缩小了搜索空间，提高搜索效率。

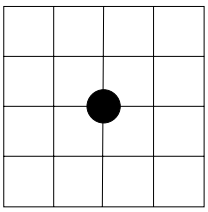


图 2　搜索范围

2.2 设置下棋风格

对一个落子估值的时候，首先在己方的角度对其进行评估，获得一个返估值 Value_Me；随后在对手的角度再进行一次评估，获得一个估值 Value_Enemy；通过 Value = K1 * Value_Me + K2 * Value_Enemy 而获得最终的估值结果。其中 K1 和 K2 是一对关键系数，当 K1 / K2 越大的时候，就表示计算机落子的攻击性更强，反之则表示计算机落子考虑较多的是阻断对方的落子，防守性更强些。K1、K2 的值可以由玩家设定，也可以由计算机根据当前棋面自己决定。与传统算法相比，这样得到的五子棋程序更加智能。

2.3 增大搜索层数

理想状态下，为了尽可能提高计算机下棋的“智力”，搜索层数应该越大越好。实际上，由于博弈树异常庞大，搜索层数的增加将会导致算法的效率大大下降。文献[4]提出多线程、分布式并行的方法。其缺点是比较消耗资源。本文算法中默认的搜索层数是 3，但玩家可以通过一个菜单选项调整搜索层数。根据已有的资源条件，最有效地进行搜索。

2.4 使用置换表

在 Alpha-Beta 搜索过程中，为了避免重复搜索已经搜索过的节点，加快搜索速度，可以使用一张表格记录每一节点的搜索结果，对任意节点向下搜索之前先察看记录在表中的这些结果。如果将要搜索的某个节点在表中已有记录，就直接利用记录下来的结果。我们称这种方法为置换表（Transposition Table,简称 TT）。

2.5 启发式搜索

五子棋游戏开局阶段有很多“定式”[5]。将“定式”的格局及其走法当成棋谱保存在数据库中，若发现当前格局存在于棋谱中，则我们不需要搜索，直接按照棋谱下棋，即利用棋谱进行启发式搜索，这样便加快了搜索的速度。该方法的关键是棋谱的模糊匹配以及棋谱的存储结构。

总结以上优化措施，如图 3 所示，我们可以得出智能五子棋博弈算法按深度优先扩展一个棋局的流程。说明：若棋局 k 是棋局 x 的某个子节点，则 g(k)表示对当前棋面 k 应用估值函数得到的估值，f(x)表示对 x 的所有子节点应用最大最小原理后得到的值。该算法有效地减少了搜索的范围，增加了程序的智能。

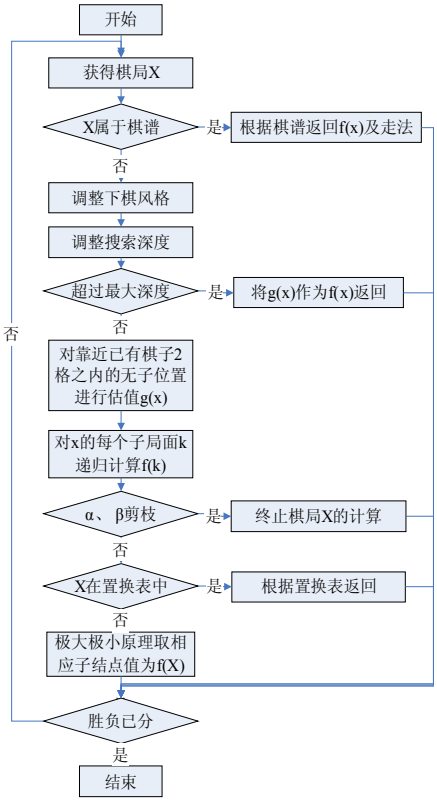
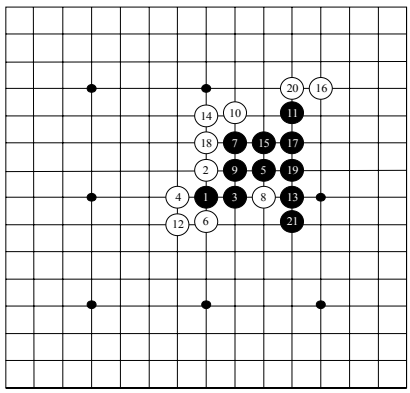


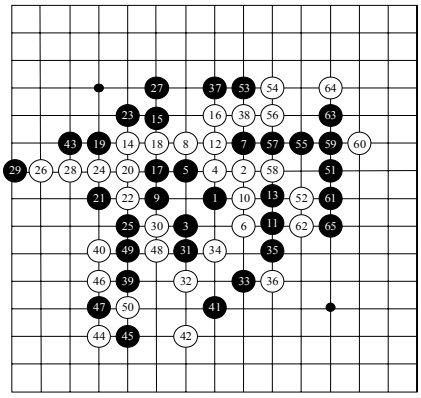
图 3　智能五子棋博弈算法流程图

3　算法实例

使用上述优化算法后，五子棋程序的智能程度和对弈水平有了明显的提高。与业余棋手对弈时，若计算机执黑先行，计算机不仅获胜的概率高，而且时间短。若玩家执黑先行，则比较难击败计算机。这两种赢棋实例的详细过程分别见图 4.a 及图 4.b。



(a) 计算机执黑先胜的情况



(b) 玩家执黑先胜的情况

图 4 计算机黑先胜(a)及其玩家黑先胜(b)

4 结论与展望

本文介绍了智能五子棋游戏的设计实现方法，提出优化五子棋算法、提高系统智能的措施。与其他五子棋程序相比，由于采用上述优化策略，本文实现的五子棋程序在对弈的水平和搜索效率方面均有显著的提高。同时，本文的设计思路也可以为其他棋类游戏算法的研究者带来启发性的思考[6,7]。下一步工作是考虑在本文算法的基础上，将五子棋国际规则中禁手、换棋等复杂规则实现，以及进一步的对弈智能化。

参考文献:

[1] 五子棋游戏介绍. 联众游戏网 [EB/OL]. (2003-3) [2007-5-25] http://www.ourgame.com/game/game-intro-new/h2g.html?gameid=10005$1

[2] 王永庆. 人工智能原理与方法[M]. 西安: 西安交通大学出版社, 1998: 288-290.

[3] Patrick Henry Winston. 人工智能[M]. 崔良沂, 赵永昌 译. 第3版. 北京: 清华大学出版社, 2005: 75-82.

[4] 许南山, 从磊, 孙风平. 并行实现有自学习能力的五子棋 AI[J]. 计算机工程与应用. 2006, 30: 45-47.

[5] 五子棋 26 种开局助记图. 中国五子棋网[EB/OL]. (2007-3-10) [2007-5-30]. http://www.wuzi8.com/Article/ Html/766.html.

[6] Victor A. Searching for Solutions in Games and Artificial Intelligence. PhD thesis [D]. Maastricht, The Netherlands: University of Limburg, 1994.

[7] Andreas J, Jonathan S. Search versus knowledge in game-playing program revisited [C]// 15th International Joint Conference on Artificial Intelligence, Proceedings Vol. I. CA, USA: Morgan Kaufmann, 1997, 692-697.

(上接第 1032 页)

用方法(15)-(16)计算分数阶初值问题(11)-(12)。取 B=0.5，C=0.5，$y(0)$=0，$y'(0)$=0，$\beta=\alpha-1$，则

$$[\alpha_1,\alpha_2,\alpha_3,\alpha_4]^T=[\alpha-1,2-\alpha,\alpha-1,2-\alpha]^T,$$ 及

$$f(t)=\begin{cases}8 & (0\le t\le 1);\\ 0 & (t>1).\end{cases}$$

计算结果如图 3 所示。

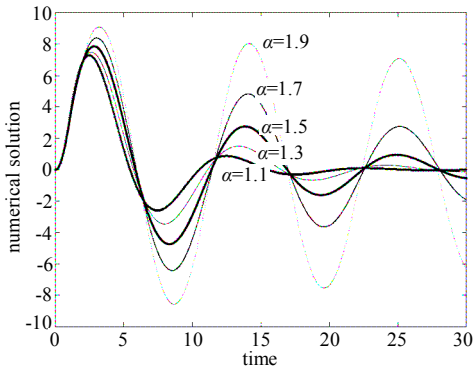


图 3 例 2 的图形 ($1<\alpha<2, h=0.025$)

该图表明了当 α 越接近 1，阻尼越大；α 越接近 2，受惯性力影响越大，这说明分数阶的模型更真实地反映了实际。经过比较，该图形与文[9]中的图形是一致的，这也说明了显式方法的有效性。另外，本例也将该算法推广到了多项高次分数阶微分方程的数值求解，与文[9]中的方法相比，方法(15)-(16)并不需要初值是齐次的，因而更适合于一般的分数阶 *Bagley-Torvik* 方程的计算。

参考文献：

[1] S Shen, F Liu. Error analysis of an explicit finite difference approximation for the space fractional diffusion equation with insulated ends [J]. ANZIAM J. (S1446-8735), 2005, 46(E): C871-C887.

[2] S B Yuste, L Acedo. An explicit finite difference method and a new von neumann-type stability analysis for fractional diffusion equations [J]. SIAM J. Numer. Anal. (S0036-1429), 2005, 42(5): 1862-1874.

[3] Luciano Galeone, Roberto Garrappa. On Multistep Methods for Differential Equations of Fractional Order [J]. Mediterr. J. Math. (ISSN: S1660-5446), 2006, 3(3-4): 565-580.

[4] K Diethelm N J Ford. Analysis of fractional differential equations [J]. J. Math. Anal. Appl. (S0022-247X), 2002, 265(2): 229-248.

[5] 王仁宏. 数值逼近 [M]. 北京：高等教育出版社, 1999.

[6] K Diethelm, N J Ford, A D Freed. Detailed error analysis for a fractional Adams method [J]. Numerical Algorithms (S1017-1398), 2004, 36(1): 31-52.

[7] I Pudlubny. Fractional Differential Equations [M]. San Diego, USA: Academic Press, 1999.

[8] C Yang, F Liu. A computationally effective predictor-corrector method for simulating fractional-order dynamical control system [J]. ANZIAM J. (S1446-8735), 2006, 47(E): 168-184.

[9] S Shen, F Liu. A computational effective methods for fractional order Bagley-Torvik equation [J]. 厦门大学学报(自科版), 2004, 45(3): 306-311.