文章编号：2095-2163(2020)06-0116-04 中图分类号：TP312 文献标志码：A

基于 UCT 算法的 Hex 棋博弈系统的研究

王 鑫，李 媛，王静文，李振宇
（沈阳工业大学 理学院，沈阳 110870）

摘 要：Hex 棋是一种基于最短路径的计算机博弈游戏，各类搜索算法均可应用于该游戏的博弈系统，针对 Hex 棋的特征提出了基于 UCT 算法的搜索方法，并与策略系统相结合对整个系统进行优化。以此算法开发的博弈系统获得了辽宁省计算机博弈大赛亚军验证了该算法的有效性。

关键词：Hex；UCT；计算机博弈；策略系统.

Hexchess game system based on UCT

WANG Xin，LI Yuan，WANG Jingwen，LI Zhengyu
（School of Science，Shenyang University of Technology，Shenyang 110870，China）

【Abstract】 Hex Chess is a kind of computer game based on the shortest path，all kinds of search algorithms can be applied to the game system，this paper puts forward the search method based on UCT algorithm for the characteristics of Hex Chess，and optimizes the whole system with the combination of the strategy system. The game system developed by this algorithm obtains the winning of the computer game competition in Liaoning Province to verify the validity of the algorithm.

【Key words】 Hex；UCT；computer game；strategy system

0 引 言

机器博弈是人工智能的一个重要的分支，主要研究对象是复杂的棋牌类游戏。计算机博弈算法的研究为人工智能利于扩充了许多新的实用算法，丰富了人工智能领域的理论成果。计算机博弈的核心技术就是在有限的时间内获得博弈的最佳结果，其技术主要包含搜索和估值，估值是对局面的一种评估，当评估方法确定后，估值优化的空间相对较小，因此，在博弈游戏的开发中搜索方法成为了计算机博弈游戏的主要研究内容。Hex 棋是近几年兴起的计算机博弈游戏是国际计算机奥林匹克锦标赛的的项目之一，自 2016 年起成为我国计算机博弈锦标赛的比赛项目。

1 Hex 棋简介

典型的棋盘由 11×11 个六边形单元格组成[1]，上下两个边界线为红色、左右两个边界线为蓝色，红色（横向）坐标表示范围 A～K，蓝色（纵向）坐标表示范围为 1～11，如图 1 所示。

Hex 棋的棋子为两种颜色的圆形棋子（略盘上的六边形单元格），红与蓝。对弈双方各执一种颜色的棋子。Hex 棋的游戏规则为：（1）比赛开始后，双方交替落子，每次只能落一个棋子，每个棋子占据一个六边形单元格。（2）两个相邻的同色棋子被认为相互连通。（3）最先将同色的两个边界用同色棋子连通的一方获胜（例如图 1 中的红方胜）。（4）不存在和棋。

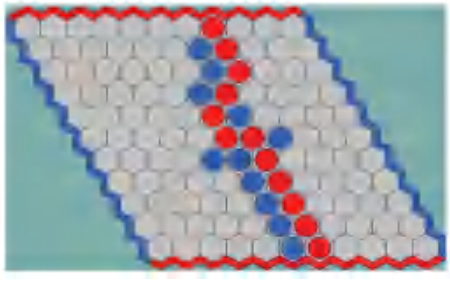

图 1 Hex 棋棋盘

Fig. 1 The board of Hex

2 算 法

计算机博弈游戏的主要核心包括搜索算法和估值，传统的搜索算法主要是极大极小算法，以及对极大极小算法的改进，这类算法都有一定的搜索深度 d，当达到搜索深度 d 时，搜索算法通过评估函数得到评估值最大分支。当前比较常用的搜索算法通常为 UCT 搜索算法[2-3]，其特点是时间或搜索次数可控，只要进行足够的模拟就可以获得准确的结果。

作者简介：王 鑫（1997-），男，本科生，主要研究方向：计算机博弈；李 媛（1976-），女，博士，教授，主要研究方向：人工智能和随机过程；王静文（1965-），男，工程师，主要研究方向：人工智能和信息安全；李振宇（1997-），男，本科生，主要研究方向：计算机博弈。

收稿日期：2020-11-26

2.1　UCT 算法

UCT 算法是一种新型的搜索算法，它是 Monte-Carlo 算法的改进。它将 Monte-Carlo 算法与 UCB 公式相结合，通过大量节点的模拟，来获得最优节点。该算法在搜索过程，针对不同的分支搜索深度可以不同，因此，可以获得更深的搜索“深度”，有效提高了搜索效率。UCT 算法的树内选择计算主要通过以下公式完成：

$$r_i = v_i + c \times \sqrt{\frac{2\ln(\sum_i T_i)}{T_i}}. \qquad (1)$$

式中：v_i 是以节点 n_i（非叶节点）为根节点的所有仿真结果的平均值，反映了根据目前仿真结果观察到的节点 n_i 能提供的回报值的期望；T_i 是节点 n_i 的访问次数，也是被树内选择策略选中的次数；c 是一个参数，用来平衡深度优先还是宽度优先。

UCT 算法的执行过程如图 2 所示。UCT 算法的 4 个过程的用途分别为：

（1）选择。从根节点出发，用递归的方法对于每一个孩子节点进行选择，选择 r_i 最大的节点作为下一次选择的开始，直到叶子节点。

（2）扩展。通过选择的的节点，将所有合法的下法作为新的叶子节点加入到搜索树中，并正确初始化 v 值和 T 值。

（3）仿真。简单的仿真策略是最对所有合法的下法，均匀的随机选择下一步，叶子节点的评估策略可以比较简单，例如：当己方获胜时为 1，对方获胜时为 0。通过若干次仿真后获得新的 v 值。

（4）反向传播。当叶子节点通过仿真获得新的 v 和 T 时，*UCT* 算法通过结果回传更新搜索路径上的所有节点 v 和 T 的值，其计算公式为：

$$T = \sum_i T_i, \qquad (2)$$

$$v = \frac{\sum_i v_i T_i}{T}. \qquad (3)$$

即父节点的访问次数 T 是所有叶子节点访问次数 T_i 之和，v 是当前节点下所有叶子节点 v_i 的加权平均值，权值为子节点的访问比例 T_i/T。结果回传从叶子节点开始，沿搜索路径逐级向上更新，直到根节点。

2.2　策略系统

策略系统主要针对 Hex 棋的一些特殊棋型[4]，这些棋型出现时可以优先考虑，并根据棋型给出相应的下棋策略以减少程序搜索的时间。策略系统主要考虑无用位置、被捕获的位置和边界形态 3 种情况：

（1）无用位置。无论哪一方行棋都不会对最终的游戏结果产生影响。当一个位置无用，无论它是由黑棋、白棋占领，或没有棋子，游戏的结果是相同的，所以我们可以任意放置棋子。图 3 所示的是相关的无用位置。

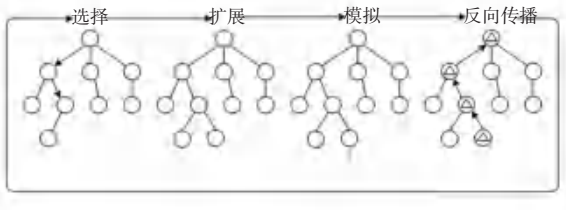

图 2　UCT 算法的基本过程

Fig. 2　The process of UCT

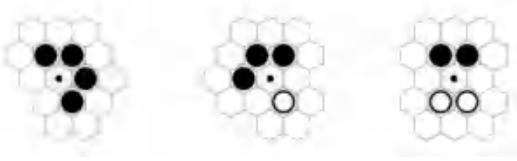

图 3　无用位置

Fig. 3　Useless position

（2）被捕获的位置。图 4 中每个形态都有两个已标记的单元格。如果白棋下在其中一个单元格，黑棋必下在另一个单元格保证周边棋子连接，形成无用模式使白棋失效，而且这两个单元格被黑色填满并不影响我们称这两个单元格被黑色捕获。形成图 4 所示形态后，黑棋方可以暂时不去考虑连接，而有限选择其它位置，当白棋方下在其中一个位置后，才需要进行连接。

图 4　标记的单元格被黑色捕获

Fig. 4　The marked hexes can be occupied by Black without changing the outcome of the game

（3）边界形态。在边界形态中，即使对手有下一步的拦截行动，它也能让自己与边界相连。例如在图 5 所示的形态中阴影表示单元格无棋子，白棋拦截，黑棋形成双桥与边界相连。

图 5　黑棋形成双桥与边界相连

Fig. 5　Black hex forms double bridges to the boundary

上述 3 种形态是 Hex 棋博弈系统中常见 3 种形态,在博弈程序中可以有效提高搜索效率。

2.3 UCT 算法改进

UCT 算法的改进主要针对 Hex 的特点进行改进[5-6],将 Hex 棋的策略系统与 UCT 算法相结合,进一步提高算法效率和准确性。

2.3.1 节点拓展方法的改进

首先对所有可下位置 M 进行无用单元格分析,对无用节点进行剪枝。下面提出 3 种拓展方案:

(1)当前节点 m_i 的模拟次数 V_i 达到规定次数 n 进行拓展, $V_i > n$.

(2)当前节点 m_i 的模拟次数 V_i 超过其兄弟节点数目的两倍再进行拓展。$k = 2$,

$$V_i > k \sum_{m \in M} V_m. \tag{4}$$

(3)当前节点的胜率大于其所有兄弟节点的胜率时进行拓展。这里我们假设 V_i 是节点的访问次数, X_i 是模拟胜率, σ_i 是所有合法节点 m_i 胜率的标准差, X_{L_i} 和 X_{R_i} 分别为 m_i 胜率的上下置信界,计算公式为

$$X_{L_i} = X_i + C\left(\frac{\sigma_i}{\sqrt{V_i}}\right), \tag{5}$$

$$X_{R_i} = X_i - C\left(\frac{\sigma_i}{\sqrt{V_i}}\right). \tag{6}$$

其中,C 设置为 1.96, C 决定了实际胜率所处的范围,1.96 胜率有 95%的概率处于这个置信区间。

假设被选中的当前节点 m_i 是它所有兄弟节点中最好的而兄弟节点 m_{second} 的胜率仅次于它,如果满足下列表达式,则 m_i 可以被拓展。

$$X_{L_i} - X_{R_{\text{second}}} \geqslant Th. \tag{7}$$

2.3.2 应用策略系统的节点模拟

在节点的模拟中,首先对用向量存储所有可下位置,再将向量中的位置随机按颜色填满整个棋盘,待棋盘填满后判断输赢。模拟时加入以下 3 种模拟策略:

(1)双桥自动连接。如图 6 所示,白棋破坏黑方双桥,黑棋必下另一端点,保证连接。模拟当前节点时,如果其父节点拦截双桥,当前节点没有做出回应保证连接,则自动进行填充将双桥破坏,再对棋盘进行模拟。

(2)被捕获区域自动填充。对要模拟的棋盘中被黑棋或白棋捕获的区域(图 4)填充对应颜色的棋子。

(3)边界形态自动行棋。如果随机模拟中形成边界模板(图 5),则在模板中自动行棋保证棋子与边界的连接。

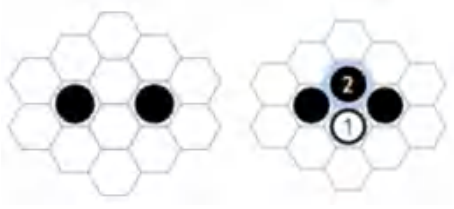

图 6 黑棋始终保证两端连接

Fig. 6 Black can always keep the two stones connected

3 实验与结果

海克斯棋可行的算法包括 Alpha-Beta 算法、UCT 算法、和结合了策略系统的改进的 UCT 算法,本文所要进行的比较主要针对以上 3 种方法进行。

为了能够获得更为有效的数据,在进行比较之前首先对 UCT 算法中的参数 c 进行优化,用 UCT 算法与两层的 Alpha-Beta 算法对比以获得最佳参数。不同节点对比的次数为 100 次,先后手各 50 次,其优化结果见图 7。

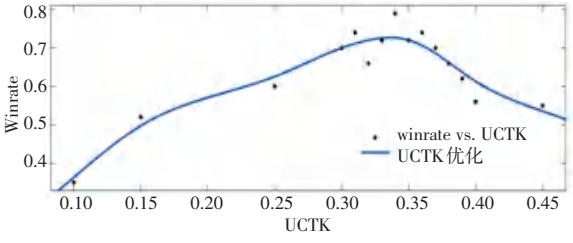

图 7 UCT 算法参数 c 的优化结果

Fig. 7 Optimization result of parameter c

根据拟合曲线选择 c 为 0.34 作为最优参数。其次对改进的节点拓展中的条件进行对比选取最佳方案。令方案 1 中 $n = 50$ 再选取不同的方案 3 中不同的 Th 进行比较结果如下图 8。

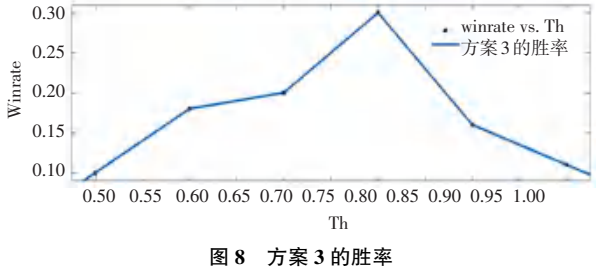

图 8 方案 3 的胜率

Fig. 8 The winning percentage of option 3

由上图可知方案 3 中 Th 为 0.8 时胜率最大,但是胜率均没超过 0.5,证明方案 1 的拓展条件优于方案 3 的拓展条件。选取方案 1 中的不同规定次数 n 与方案 2 进行对比,对比结果见图 9。

由上图可知不同规定次数 n 对比方案 2 胜率均在 0.5 左右,当 n 取 60 时胜率达到 0.57 最优。以

c = 0.34 为 UCTK 参数对比完全改进(应用模拟中的策略系统)前后的 UCT 算法过程中先后手分别对弈 100 局,对比的结果见表 1。

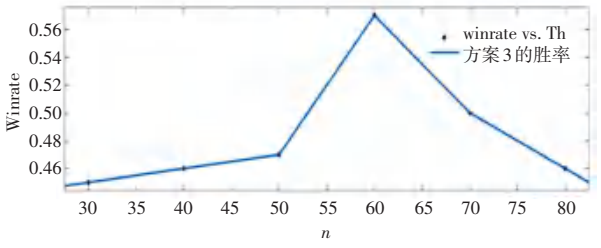

图 9　方案 1 的胜率

Fig. 9　The winning percentage of option 1

表 1　改进前后 UCT 的对比结果

Tab. 1　Comparison of different UCT algorithm

对弈局数	改进后胜局数	胜率/%	先后手
100	86	86	后手
100	93	93	先手

从表 1 的结果可知,在具有相同的 UCT 算法参数 c 的条件下,含有策略系统的改进的 UCT 算法与改进前的 UCT 相比获得较高的胜率,同时,无论是先手还是后手,改进的 UCT 算法的胜率都要大于 0.5。与单一的 UCT 算法相比,含有策略系统的改进 UCT 算法具有更高的胜率。

4　结束语

利用 UCT 算法结合棋型策略系统可以有效提高 Hex 棋人工智能系统的水平,有效降低了最优位置的搜索次数和搜索时间,极大提高了分支选择的效率。以该算法开发的 Hex 棋博弈程序获得了 2019 年辽宁省本科大学生计算机博弈竞赛 Hex 棋亚军及 2019 年全国大学生计算机博弈大赛 Hex 棋亚军,进一步验证了该算法的有效性。

参考文献

[1] 计算机博弈大赛. Hex 棋游戏规则[EB/OL]. http://computergames.caai.cn/jsgz19.html.

[2] KLOETZER J. Monte Carlo Opening Books for Amazons: CG2010, Verlag Berlin Heidelberg, 2011[C]. Springer.

[3] GELLY S, SILVER D. Monte-Carlo tree search and rapid action value estimation in computer Go[J]. Artificial Intelligence, 2011(175):1856-1875.

[4] HUANG S, ARNESON B, HAYWARD R B, et al. MoHex 2.0 a pattern-based MCTS Hex player: International Conference on Computers and Games, Yokohama, Japan, 2013[C]. Springer.

[5] BRODERICK ARNESON R B H P. Monte Carlo Tree Search in Hex[J]. Trans. on Comput'l Intel. and AI in Games, 2010, 2(4):251-257.

[6] RASMUSSEN RUNE M F. A Move Generating Algorithm for Hex Solvers: AI2006, Verlag Berlin Heidelberg, 2006[C]. Springer.

(上接第 115 页)

从图 6、7 可知,在 X-DSP 存储空间处于保护状态(CSM=1)和非保护状态(CSM=0)两种情况下完成了存储保护单元的功能测试。表明存储保护单元实现了 X-DSP 存储空间的数据保护功能,阻止不具有访问权限的用户访问存储保护区域,同时产生指令预取中止和数据访问中止,并断开仿真器连接。

4　结束语

本文针对 X-DSP 存储空间的访问安全问题,基于硬件保护方法完成了存储保护单元的 RTL 代码设计,并采用 System Verilog Assertions 编写存储保护单元的功能属性描述,用断言验证方法完成了存储保护单元的功能验证。同时,在 X-DSP 芯片验证环境下采用 FPGA 原型验证,完成了存储保护单元的功能测试。结果表明,所设计的存储保护单元实现了 X-DSP 存储空间的数据保护功能。此外,断言验证方法不仅保证了功能验证的完备性,还缩短了产品开发周期。

参考文献

[1] 刘明. 面向多核 DSP 的 DMA 访问数据一致性优化和验证平台研究[D]. 国防科学技术大学,2016.

[2] FIORIN L, PALERMO G, LUKOVIC S, et al. Secure memory accesses on networks-on-chip[J]. IEEE Transactions on Computers, 2008, 57(9): 1216-1229.

[3] AHN D, LEE G. A memory-access validation scheme against payload injection attacks[J]. IEEE Transactions on Dependable and Secure Computing, 2014, 12(4): 387-399.

[4] 朱车壮, 陈岚, 冯燕. 基于覆盖率驱动的 SoC 验证技术研究[J]. 微电子学与计算机, 2011, 28(11):48-52.

[5] 朱峰,鲁征浩,朱青. 形式化验证在处理器浮点运算单元中的应用[J]. 电子技术应用, 2017, 43(2): 29-32.

[6] 李黎明,关永,吴敏华,等. 运用定理证明的形式化方法验证 SpaceWire 编码电路[J]. 小型微型计算机系统, 2012, 33(6): 1372-1376.

[7] CHEN W, RAY S, ABADIR M, et al. Challenges and Trends in Modern SoC Design Verification[J]. IEEE Design & Test, 2017:7-22.

[8] PORTILLO J, JOHN E, NARASIMHAN S. Building trust in 3PIP using asset-based security property verification[C]//2016 IEEE 34th VLSI Test Symposium (VTS). IEEE, 2016: 1-6.

[9] SHIRAZ S, HASAN O. A library for combinational circuit verification using the HOL theorem prover[J]. IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, 2017, 37(2): 512-516.

[10] 陈钢,于林宇,裘宗燕,等. 基于逻辑的形式化验证方法:进展及应用[J]. 北京大学学报(自然科学版), 2016, 52(2): 363-373.