杨鑫朋, 王静文. 基于改进 UCT 算法的苏拉卡尔塔棋理论研究[J]. 智能计算机与应用,2025,15(12):83-87. DOI:10.20169/j.issn.2095-2163.24041003

基于改进 UCT 算法的苏拉卡尔塔棋理论研究

杨鑫朋, 王静文
(沈阳工业大学 理学院, 沈阳 110870)

摘 要: UCT(Upper Confidence Bound Apply to Tree)算法,即上限置信区间算法,是对蒙特卡洛算法利用 UCB1 算法进行改良的随机模拟算法。但对于苏拉卡尔塔棋,有大量的棋子数较多的棋局,单次模拟的随机性很难快速完成一次有效对局。甚至过多随机模拟会使单局的最终结果与真实情况大相径庭。对此利用限制单局模拟次数搭配估值函数来改进 UCT 的模拟函数,有效的提升了 UCT 算法的模拟速度和模拟准确性,提高了 UCT 算法的博弈能力。

关键词: 苏拉卡尔塔棋; UCT 算法; 改进 UCT 算法
中图分类号: TP312 **文献标志码**: A **文章编号**: 2095-2163(2025)12-0083-05

Research on the theory of Surakarta chess based on improved UCT algorithm

YANG Xinpeng, WANG Jingwen
(School of science, Shenyang University of Technology, Shenyang 110870, China)

Abstract: UCT (Upper Confidence Bound Apply to Tree) algorithm is a stochastic simulation algorithm modified by Monte Carlo algorithm using UCB1 algorithm. However, in the case of Surakarta, there are a large number of chess games with a large number of pieces, and the randomness of a single simulation is difficult to quickly complete an effective game. Even too many random simulations can make the final result of a single round very different from the real situation. In this regard, the simulation function of UCT is improved by limiting the number of simulations in a single game and the estimation function, which effectively improves the simulation speed and accuracy of the UCT algorithm and improves the game ability of the UCT algorithm.

Key words: Surakarta; UCT algorithm; improvement UCT algorithm

0 引 言

苏拉卡尔塔棋起源于印度尼西亚爪哇岛,是一种古老的双人对弈棋种。其原名为 Permainan,在印尼语中为“游戏”的意思。后因其发源地名为苏拉卡尔塔,因此被法国人称为 Surakarta 棋。

在中国,2010 年全国大学生计算机博弈竞赛[1]开始将该项目作为竞赛项目之一,随着国内外各种机器博弈类竞赛的开展,苏拉卡尔塔棋也逐渐走进大众视野,UCT 算法成为博弈系统普遍使用的算法之一,但在苏拉卡尔塔棋上直接使用 UCT 算法暴露出了一些问题,本文对其存在的问题进行研究并优化了 UCT 算法。

1 Surakarta 棋简介

1.1 棋盘介绍

苏拉卡尔塔棋为 6×6 的正方形棋盘,总计 36 个棋位。如图 1 所示为棋位编码,其中有 8 段圆弧,分别衔接棋位 $c5-a3$, $d5-f3$, $c0-a2$, $d0-f2$, $b5-a4$, $e5-f4$, $b0-a1$, $e0-f1$。

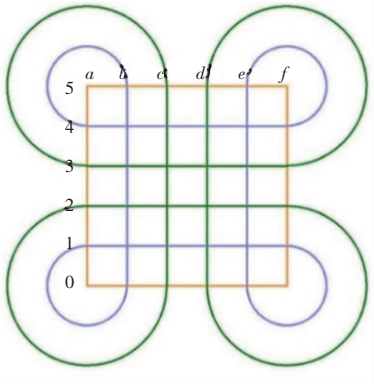

图 1 Surakarta 棋棋盘
Fig. 1 Chess board of Surakarta

作者简介: 杨鑫朋(2003—),男,本科生,主要研究方向:计算机博弈。
通信作者: 王静文(1965—),男,学士,工程师,主要研究方向:人工智能和信息安全。Email:wangjingwen007@126.com。
收稿日期: 2024-04-10

1.2 Surakarta 棋基本规则

开局时,在棋盘对侧各放置 12 颗棋子构成开局局面(在本研究中,将棋子分为红棋与黑棋,并将游戏双方称为黑方与红方)。双方选手由掷硬币的方式来决定优先行棋方。

行棋[1-2]时,在不考虑吃棋的情况下,一个棋子可以有 8 个方向(上,下,左,右,左上,左下,右上,右下)行进一个空的棋位。若想进行吃棋行为,需要经过一个完整的弧,且经过的路径中,不能有棋子阻挡。

2 Surakarta 棋相关算法

2.1 走法生成函数的实现

传统的走法生成[3-5]是引入拓展棋盘,在拓展棋盘对应的轨道变换处标记连接的轨道入口坐标,再进行重复遍历,判断。需要将 6×6 数组增广为 8×8 数组,增加了遍历过程中的时间消耗。并在获取可吃棋子位置时,还需要判断是否已经通过一段轨道,使得吃棋位置的获取比较复杂。

而如果直接将外轨道与内轨道的所有位置顺逆时针飞行的轨道用如下式所示的二维数组存储起来,在判断是否能够吃棋时,仅需要通过遍历数组,判断到吃子位置之前是否遇到阻碍,以及 state 是否为 1 即可,由于仅需要从位置以及状态两个维度即可实现,因此使用二维数组存储。

$$\mathrm{Orbit}[23][2] = \{\bigcup_{k=1}^{23}\{i_k \times 6 + j_k, \mathrm{state}\}\} \quad (1)$$

k 表示顺逆时针遍历轨道时对应位置的序号。i 与 j 表示棋子在棋盘中的轨道上对应位置的行数与列数(从 0 开始)。 state 来标记当前轨道位置是否已经经过圆弧,未经过标记为 0,经过标记为 1。

如 $b5$ 位置对应的顺时针遍历轨道利用公式(1)获得的计算值为:

Orbit1[23][2] = { {7, 0},{13, 0},{19, 0},{25, 0},{31, 0},{24, 1},{25, 1},{26, 1},{27, 1},{28, 1}, {29, 1},{34, 1},{28, 1},{22, 1},{16, 1},{10, 1},{4, 1},{11, 1},{10, 1},{9, 1},{8, 1},{7, 1},{6, 1} }。

2.2 Alpha-beta 搜索算法

Alpha-beta 搜索算法[6-7]是对极大极小搜索算法的改良,主要对生成树的一些非必要结点进行剪枝操作,减少搜索结点数。其主要剪枝操作如图 2 所示。

在进行 c 结点的估值返回时,先搜索 d 结点,再搜索 e 结点,由于 c 结点处于 min 层,且 e 结点估值小于 d 结点估值,因此将 e 结点估值 2 返回给 c 结点。在此时,由于 a 结点在 max 层,应当选取 b,c 结点中的最大值。c 结点处于 min 层,后续返回的估值只会比 2 小,且 b 结点的估值此时已经大于 c 结点估值,因此 c 结点下未进行搜索的子节点对于 a 的值不再产生影响,可以跳过搜索过程。这个过程被称之为剪枝。

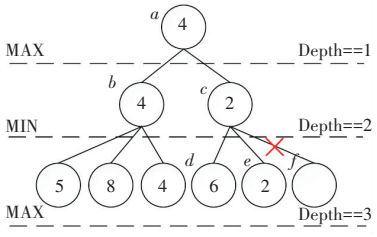

图 2 Alpha-beta 剪枝算法实例

Fig. 2 Example of Alpha-beta algorithm

2.3 Alpha-beta 类搜索算法问题概述

此类算法的主要特点是依靠估值函数对局面反馈的准确性,从而精准获得当前局面下的最优下棋位置。

由于要进行全盘搜索,搜索深度较大时搜索树巨大,由于博弈竞赛的时效性,即使搭配剪枝也无法在较短的时间内达到一个较深程度的搜索,因此无法将估值的优势发挥到最大。

2.4 UCT 算法(上限置信区间算法)

UCT(Upper Confidence Bound Apply to Tree)算法[6, 8-9]是对蒙特卡洛算法(MCTS)[10-12]的结点选取阶段用 UCB1 算法(Upper Confidence Bound)[13]的思想进行改良。蒙特卡洛算法的弊端在于需要大量模拟数据的支撑。在改良后,提高了算法对数据的利用率。在较少的模拟次数下也可以实现取优效果。其改良的实际操作是对结点的模拟胜率利用 UCB 公式进行计算,使得在结点的选取上实现 UCB1 算法思想。UCB 计算公式[6]如下:

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

其中 r_i 代表结点 i 胜率的计算值;v_i 代表从结点 i 出发返回的原始胜率值;c 是平衡探索与利用的系数,要进行取优操作来获得模拟对象的最佳 c 值;T_i 表示节点 i 的模拟总数;$\sum_i T_i$ 是同层节点的模拟总数。这样随着同层节点的模拟次数增加,模拟次数较少的节点会显露优势,避免拓展的过分倾斜。

UCT 算法的执行过程[6]如图 3 所示:

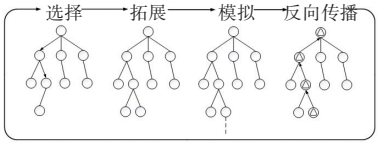

图 3　UCT 算法执行过程

Fig. 3　Execution process of UCT

2.5　UCT 算法 c 值获取

由于 Alpha-beta 算法的稳定性，这里选取深度为 3 的 Alpha-beta 搜索算法与不同 c 值的 UCT 先后次序对战 100 次[14]，实验数据如图 4 所示。数据采用 degree 为 8 的 Polynomial 进行插值拟合，获得的最优 c 值约为 0.23。

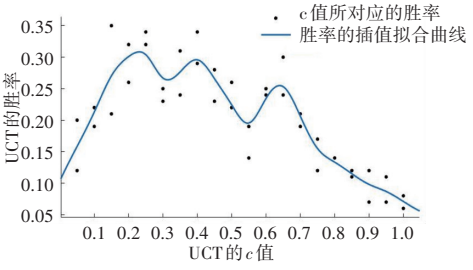

图 4　UCT 算法 c 值优化图

Fig. 4　UCT algorithm c optimization diagram

2.6　UCT 算法问题概述

UCT 算法的主要环节分别是选择与模拟。应当选择一个合适的结点；尽可能真实地反应此结点对于全局的影响，并且尽可能获得较多模拟数据来校准随机模拟带来的偏差。

对于选择阶段，UCB1 计算公式已经对其进行了足够好的优化。因此考虑在模拟时可以在相同时间内给出较好的模拟数据以及进行足够多的模拟次数。

3　Surakarta 棋算法改进

由于 Alpha-beta 搜索算法的估值函数能够有效地反应局面状态，基于 UCT 算法所存在的问题，在此考虑是否可以将估值函数与 UCT 的模拟进行结合，利用估值函数的局面反馈来降低 UCT 算法模拟的随机性，提高模拟结果的准确性。

3.1　估值函数

估值函数[15]是通过对局面信息的计算获得一个数值，通过该数值的大小可以判断出局势的利弊。

利用估值矩阵，剩余棋子数目以及特殊布局获得局面估值来尽可能真实的反应局面状态。估值矩阵可以有效控制棋子布局，棋子的剩余数目可以反馈局势的好坏，特殊布局可以牵制对手在未来的几个局面中处于劣势。

估值矩阵[16]：

$$\text{Matrix\_V}=\begin{pmatrix}5&20&20&20&20&5\\20&30&50&50&30&20\\20&50&40&40&50&20\\20&50&40&40&50&20\\20&30&50&50&30&20\\5&20&20&20&20&5\end{pmatrix}\tag{3}$$

棋子估值：定义变量 Pawn_V 记录棋子的估值大小，每遇到黑色棋子估值加 100，遇到红色棋子时，估值减 100。

特殊布局：三手进攻[4]

如图 5 所示，在此局面下黑棋已经完成了三手进攻布局，在 $b4$ 棋位的黑棋可以优先对红棋发起进攻，使黑棋始终胜红棋一子。由于在开局局面下（以黑棋为例）$b4-c3$，$a4-a3$，$b5-b4$ 三步即可形成此局面，因此被称之为三手进攻。

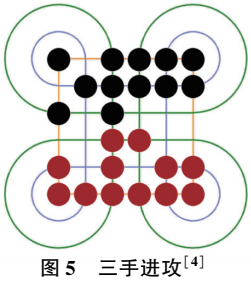

图 5　三手进攻[4]

Fig. 5　Three-step attack[4]

在搜索深度≥7 时，由于三手进攻是先手情况下唯一一种在 7 步内能对对方造成打击的棋形，剪枝算法只是对全搜索算法搜索效率的提升，不会影响最优节点的选取，因此在估值函数中只要棋子估值大于估值矩阵中的最大值，一定会优先考虑吃掉对方棋子，因此会自动形成三首进攻的局面。

估值函数伪码如下，返回值的正负反应游戏方的利弊，其中 BLACKPAWN 代表黑方棋子，REDPAWN 代表红方棋子。

```
START
Function value(board)
  value = 0
  For (board)
      If (board[i][j] == BLACKPAWN)
          value -= Matrix_V[i][j]
          value -= Pawn_V
      else If (board[i][j] == REDPAWN)
          value += Matrix_V[i][j]
          value += Pawn_V
  return value
END
```

3.2 优化 UCT 模拟过程

改进 UCT 算法流程图如图 6 所示,其中被标记区域为对 UCT 算法的改进措施。

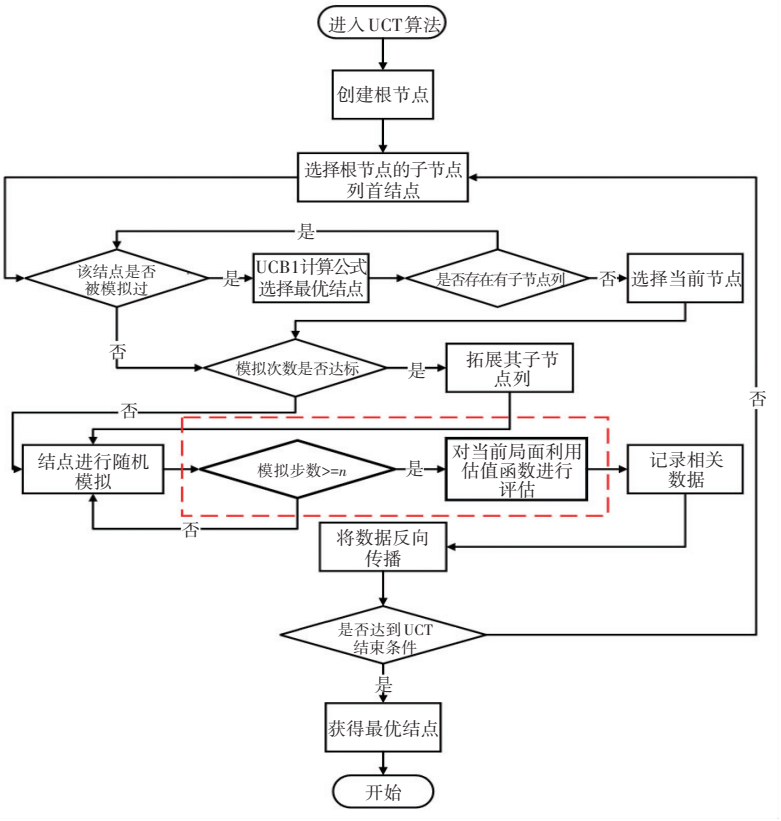

图 6 改进 UCT 算法流程图

Fig. 6 Improved UCT algorithm flowchart

在原始的 UCT 模拟过程中,需要模拟到局面结束来获得胜负。由于模拟的随机性,平均一局要进行 230 步以上。而在正常的对弈中一局的行棋大致在 50 步之内,过多的随机模拟会导致结果的偏差过大,需要大量的对局数才能有效校准。在相同时间内,单局步数与总模拟次数呈现反比关系,因此想要结果准确需从以下两方面考虑:

(1)增加在相同时间下的模拟对局数。

(2)提高单次随机模拟的准确性。

对此利用估值函数来计算第 n 步(n 由实验获取)的局面估值,当估值为负时,认定黑方获胜,反之,则认定为红方获胜。这样可以在提高模拟准确性的同时增加相同时间下的模拟对局数。

除此之外,还可以利用 OpenMP 并行计算[17],增加模拟次数以及将程序运行模拟更改为 Release 模式进行模拟提升程序对硬件的利用率,增加模拟次数。

3.3 模拟最大次数 n 值获取

选取深度为 3 的 Alpha-beta 搜索算法与不同 n 值,由于常规对局在 50 步以内,因此这里 n 值选取[0,60]的 UCT 先后次序对战 100 次,获得最优 n 值,实验数据如图 7 所示。数据采用 degree 为 4 的 Polynomial 进行插值拟合,获得的最优 n 值约为 9。

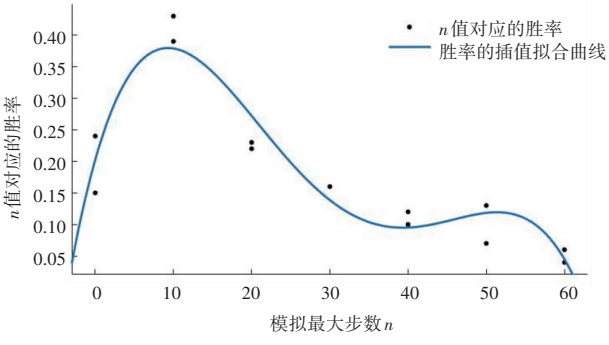

图 7 最优 n 值优化图

Fig. 7 Optimal n optimization diagram

4 实验与结果

4.1 实验环境

仿真环境:Window 10 操作系统 ; Visual Studio 2022(社区版);EasyX 图形库。

硬件环境: Intel(R) Core(TM) i7-11800H,主

频：2. 30 GHz；内存：16 GB；显卡：GeForce RTX 3050；八核十六线程。

相关调节：实验中双方均使用 openMP 并行计算（16 线程）以及 Release 模式进行实验。

4.2　实验结果

实验过程为双方分先后手分别对弈 100 局获得双方胜率。由上述实验数据可知，通过限制单次模拟的行棋步数，以及通过估值函数返回值来反应模拟结果的形式可以有效的增强 UCT 算法在 Surakarta 棋上的博弈强度见表 1～表 3。

表 1　UCT 与 估值优化 UCT 算法对比

Table 1　Comparison between UCT and UCT with value　%

先后手	UCT	估值优化 UCT
先手	22	74
后手	26	78

表 2　改进 UCT 与 改进 UCT

Table 2　Improve UCT vs improve UCT　%

先后手	改进 UCT	改进 UCT
先手	41	59

表 3　Alphabeta 与改进 UCT

Table 3　Alphabeta vs improve UCT　%

先后手	Alphabeta	改进 UCT
先手	19	81
后手	23	77

5　结束语

本文中对 Alpha-beta 搜索算法以及 UCT 模拟算法存在的问题进行了讨论，并对 UCT 算法的 c 值在苏拉卡尔塔棋的应用上进行了取优处理。同时对 UCT 算法存在的问题进行了深入研究。并利用限制单局模拟步数，搭配估值函数优化局面评估的方式，改进 UCT 算法的模拟阶段。有效地解决了 UCT 算法所存在的问题。提升了利用 UCT 算法来实现 Surakarta 棋博弈程序的博弈强度。除此之外，还可以通过对开局与残局的优化[18-20]来提升博弈系统强度。

参考文献

[1] 全国计算机博弈大赛[EB/OL]. (2022-04-09)[2024-3-31]. http://computergames.caai.cn/

[2] 张涛，江业峰，李博文. 基于 PVS 算法的苏拉卡尔塔棋博弈系统设计与实现[J]. 信息与电脑(理论版)，2023,35(19):46-48.

[3] 车晓菲，徐勇，蒋宗华. 苏拉卡尔塔棋系统的设计与实现[J]. 信息与电脑(理论版)，2021,33(6):70-73.

[4] 李东轩，胡伟，王静文. 基于 Alpha-Beta 算法的苏拉卡尔塔棋博弈系统研究[J]. 智能计算机与应用，2022,12(2):123-125.

[5] 张利群. 实现苏拉卡尔塔棋网络博弈平台的吃子算法[J]. 计算机工程与应用，2016,52(7):62-66.

[6] 王静文，李媛，邱虹坤，等. 计算机博弈算法与编程[M]. 北京：电子工业出版社，2021.

[7] KNUTH D E，MOORE R W. An analysis of alpha-beta pruning[J]. Artificial Intelligence，1975,6(4):293-326. DOI:10.1016/0004-3702(75)90019-3

[8] HUTCHISON D，KANADE T，KITTLER J，et al. Ananalysis of UCT in Multi-player Games[J]. ICCA Journal，2008,31(4):195-208.

[9] STURTEVANT N. An analysis of UCT in multi-player games[J]. ICGA Journal，2008,31:195-208. DOI:10.3233/ICG-2008-31402

[10] MÉHAT J，CAZENAVE T. A parallel general game player[J]. KI-Künstliche Intelligenz，2011，25(1)：43-47.

[11] PEREZ D，SAMOTHRAKIS S，LUCAS S. Knowledge-based fast evolutionary MCTS for general video game playing[C]// Proceedings of 2014 IEEE Conference on Computational Intelligence and Games. Piscataway，NJ:IEEE,2014:26-29.

[12] 王仁泉，丁濛，李淑琴，等. 基于强化学习的苏拉卡尔塔博弈算法[J]. 智能计算机与应用，2020,10(4):6-8.

[13] POWLEY E J,WHITEHOUSE D，COWLING P I. Bandits all the way down：UCB1 as a simulation policy in Monte Carlo Tree Search[C]// Proceedings of 2013 IEEE Conference on Computational Inteligence in Games (CIG). Piscataway，NJ：IEEE，2013:11-13.

[14] 徐志凡，王静文，李媛. 基于 UCT 算法改进的 Hex 棋博弈系统研究[J]. 智能计算机与应用，2022,12(3):183-185.

[15] 王正志，肖齐英. 博弈树搜索与静态估值函数[J]. 计算机应用研究，1997,14(4):76-78.

[16] 李淑琴，李静波，韩裕华，等. 苏拉卡尔塔博弈系统中评估函数的研究[J]. 北京信息科技大学学报(自然科学版)，2012,27(6):42-45.

[17] 侯鑫磊. 并行计算在计算机博弈中的研究与应用[D]. 重庆：重庆理工大学，2015.

[18] 张博，李淑琴，李臻. 苏拉卡尔塔棋中残局的优化[J]. 智能计算机与应用，2017,7(1):83-85.

[19] 靳淑娴，高铭，王修锴. 开局库在点格棋计算机博弈系统中的应用[J]. 数字技术与应用，2022,40(1):61-63. DOI:10.19695/j.cnki.cn12-1369.2022.01.20

[20] 魏钦刚，王骄，徐心和，等. 中国象棋计算机博弈开局库研究与设计[J]. 智能系统学报，2007(1):85-89.