文章编号：2095-2163(2021)02-0097-04 中图分类号：TP312 文献标志码：A

基于 PVS 算法的六子棋博弈系统的研究

王鸿菲，王静文，李 媛
（沈阳工业大学 理学院，沈阳 110870）

摘 要： 针对六子棋比赛中基于棋型分析的评估函数比较复杂，因此搜索效率大大降低，六子棋是一种复杂度与象棋相当的博弈游戏。其复杂性主要是平均分枝因子大，导致博弈树搜索的深度太浅。本文采用了 PVS 搜索算法，通过缩小搜索范围，从而有效增加剪枝效率，同时结合了迭代深化和历史启发增强及置换表和哈希表技术，极大提高了搜索效率和深度。使用该技术开发的六子棋系统，其博弈水平得到了有效提高。

关键词： 六子棋；PVS；路；历史启发增强；迭代加深；置换表

Connect6 based on PVS algorithm

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

【Abstract】 The evaluation function based on chess type analysis is complex in the game of Connect6, so the search efficiency is greatly reduced. The complexity is mainly due to the large average branching factor, which leads to the shallow depth of game tree search. In this paper, PVS search algorithm is used to reduce the search scope, so as to effectively increase the pruning efficiency. At the same time, the combination of iterative deepening , historical heuristic enhancement, replacement table and Hash table technology greatly improve the search efficiency and depth. The game level of Connect6 system developed by this technology has been effectively improved.

【Key words】 Connect6; PVS; road; historical heuristic enhancement; iterative deepening; Hash

0 引 言

机器博弈是人工智能研究中极具挑战性的重要课题之一，其技术进步则为人工智能领域的研究融汇了为数可观的理论和方法，对社会及学术方面产生了广泛而深远的影响。机器博弈并不是如人们设想的那样只是人和计算机间简单的下棋小游戏，而是通过这种竞争性的活动来检验人工智能成果是否达到了人的智能水平。在社会的日常生活中经常面临着决策问题，而建立和选择决策的过程便是计算机博弈研究关注的内容，所以在智能决策、沙盘推演等场景中均有着现实应用意义。

由于六子棋的棋盘与围棋棋盘相同[1]，并有与围棋有着相同的状态空间复杂度，吸引了越来越多的计算机博弈爱好者对六子棋的关注。本文拟对此展开研究论述如下。

1 六子棋简介

六子棋是近些年新兴起的棋类博弈项目，是国立交通大学吴毅成教授提出的“连 K”系列棋之一[2]。对其特点可概述为：

（1）规则简单，六子棋有黑白两方，黑方先落子。黑方第一步下一颗子，随后黑白双方轮流落两子。连成六子或以上获胜。没有禁手，长连即连成六子以上也为赢，如果棋盘被下满时仍未分出胜负则算和棋，若不同意和棋，也可按照存在五连的数量来定胜负，减少和棋局面的出现。六子棋的标准棋盘如图 1 所示。

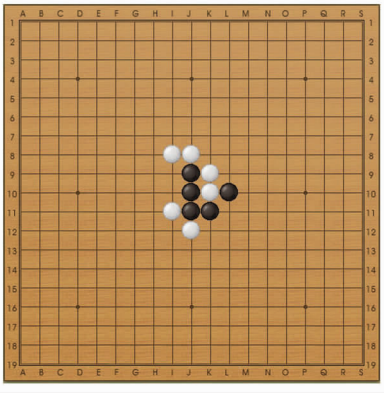

图 1 六子棋棋盘

Fig. 1 The chess board of Connect6

（2）变化复杂。对于机器博弈来说，一般采用状态空间复杂度和博弈树复杂度来衡量某种博弈游

作者简介：王鸿菲(1999-)，男，本科生，主要研究方向：计算机博弈；王静文(1965-)，男，工程师，主要研究方向：人工智能和信息安全；李 媛(1976-)，女，博士后，教授，主要研究方向：人工智能和随机过程。

收稿日期：2020-11-15

戏的复杂程度。状态空间复杂度,指的是从游戏最开始的状态可以变化出的符合规则的状态的数量[3]。在六子棋的对弈过程中的每一个局面都对应一个节点,而一个节点下面又有很多子节点,所以不断地向下推演就可以建立整个博弈树,直至得到博弈结果。但得到的博弈树是巨大的,随着不断地加深,子节点将以几何方式上升。常见棋类的复杂度对比见表 1。

表 1 常见游戏的复杂度

Tab. 1 Complexity of computer game

棋类项目	状态空间复杂度	博弈树复杂度
国际象棋	10^{46}	10^{132}
中国象棋	10^{48}	10^{150}
十九路围棋	10^{172}	10^{360}
五子棋	10^{105}	10^{70}
六子棋	10^{172}	$10^{140} \sim 10^{188}$

从表 1 中可以看到,六子棋比五子棋的复杂度高[4-5],和象棋差不多或略高一点。

(3)游戏公平。由于各方每次下完一手后,盘面都比对方多一子,因此赛局自然达成平衡的状态,这大大提升了六子棋的公平性。不像许多棋种、如五子棋、象棋、国际象棋,先下者会占据一些优势。

2 六子棋博弈系统

计算机博弈游戏的核心由搜索和估值两部分组成。其中,估值是用于准确地评价当前局面,而搜索则是根据当前局面获得最佳下法。这里将给出探讨分述如下。

2.1 估值函数

在博弈比赛的对弈中,如果将棋局的所有状态都列举出来,就一定可找到最佳的走法,但是对于博弈来说,这种做法不仅是无意义的,对于大部分棋种也是不可行的。因此对博弈树的穷举搜索必须适可而止,由此可知研究中搜索的深度是有限的,即可根据在一定深度处的节点的估计值来评分,就是估计值代替实际的搜索,这就叫做估值函数。不管对黑方、还是白方,都是利于对博弈树进行搜索找到最佳走法的。如果不能穷举所有走法,就只能在搜索到一定层后,根据对局面的估值来判断路径的好坏,此时就要设计出评估函数来对局面进行估值。估值方法往往和具体的棋类规则结合紧密,这在很大程度上决定了博弈程序的棋力高低[6]。研究时,既可以向估值函数写入棋类知识,使程序对于局面的评估更为精确,也可以写出简化的估值函数,使估值的过程简捷、且节省运算时间,期望通过更深的搜索提高棋力。

目前,六子棋普遍采用 2 种估值方式。一种是基于“路”的方式进行估值,另一种就是基于“棋型”的方式进行估值。这里,基于“路”或基于“棋型”,则是指根据路或棋型的方式对棋盘进行扫描。

需要指出的是,在基于棋型的估值中,由于目前常见的 10 种棋型都是通过经验总结的,主观因素影响较大,可能存在未定义的其他棋型。同时因为同一种棋型也有多种可下位置。

所以棋形判断的复杂度高,计算量大,受计算机博弈比赛中六子棋博弈时间的约束,搜索的深度和宽度都会受到限制。故而本文提出基于路的估值方法,相对于基于棋型的估值较为简单,实现上较为容易,搜索时间也较短,更适合在在博弈比赛中付诸应用。

基于路的估值,在一个棋盘中,将路分为 6 种,扫描连续的同一条直线上的 6 个不同的位置,再对同一个颜色的棋子进行计数,从而判断为几路。因此可以计算出一个棋盘有:水平方向为 $19\times14=266$;竖直方向为 $19\times14=266$;左斜方向 $14\times14=196$;右斜方向 $14\times14=196$,所以一共有 294 路。本文采用了 4 个方向的函数实现路的扫描。研究后推得 *AnalysisRight* 主要伪码可表示如下:

```
Int AnalysisRight(char position[ ][19], int i, int j){
    char tempArray[19];
    int x;
    int y;
    int realnum;
    if(18 - j < i){
        y = 18;
        x = j - 18 + i;
    }
    else{
        x = 0;
        y = i + j;
    }
    int g = 0;
    for(int k = 0; k < 19; k ++){
        if(x + k > 18 || y - k < 0){
            break;
        }
        tempArray[k] = position[x + k][y -
```

```
k];
            g ++;
        }
        AnalysisLine(tempArray,g,y - j);
        for(int s = 0; s < g;s ++){
            if(m_LineRecord[s] ! =
WEIFANGWEN) {
                TypeRecord[x + s][y - s][3] =
    m_LineRecord[s];
            }
        }
        return TypeRecord[i][j][3];
    }
```

其它各个方向上的计算方法和上述伪码的计算方法相同。文中不再做过多赘述。

2.2　基于 PVS 的搜索算法

六子棋的复杂度高，一般的搜索算法难以达到较高的搜索效率。目前，常用的方法有 alpha-beta 算法、UCTS 算法等[7]。其中，alpha-beta 算法在经过剪枝处理后，却仍然还是存在大量的节点需要去遍历，搜索效率较低；UCTS 算法可以达到较好的搜索深度，但得到的估值准确性较低。

本文的搜索算法是基于 PVS 算法。PVS 算法，也称最小窗口搜索算法，是由 alpha-beta 变形而来。两者间的主要区别就在于：除了主变量外的其它节点都进行零窗口搜索，并且把 α 的值复制为 β 的值，通常情况下都是把每个节点的所有子节点进行排序，同时假设第一个节点是最好的，作为主变量，进行全窗口搜索。通过零窗口搜索，判断是否存在 $\alpha \geq \beta$ 的值，在此基础上进行博弈树的剪枝。其中，对节点进行排序是该算法至为关键的一步，这样大大提高了搜索效率。所以本文采用了置换表和哈希表、历史启发增强、迭代深化等方法，极大地提高了算法的搜索效率[8]。

在搜索过程中，还增加了置换表和哈希表，如果计算过的价值，将会加入 Hash 表，并通过查找的方法判断是否存在，如果存在就直接调用价值，这样也大大加快了搜索过程；采用了历史启发方法增强通过已有的节点评分可更好地进行排序[9]，价值高的节点排在前面，极大提高了搜索效率；与此同时，还通过迭代深化来加快节点排序的过程，过程中也一并进行时间的控制。在此基础上，改进的 PVS 算法的伪码参见如下。

```
Function PVS(int depth, int alpha, int beta) {
    if(depth < = 0)
    return value()
      generateAllMove()
    sort()
    makeMove()
      best = - PVS(depth - 1, - beta, - alpha)
      unMakeMove()
      for(move i = 1 to move.size())
      if(best < beta)
        if(best > alpha)
          alpha = best
      makeMove()
      v = - PVS(depth - 1, - alpha - 1,
- alpha)
        if(v < beta && v > alpha)
          best = - PVS(depth - 1, - beta, - v)
        if(v > best)
          best = v
        unMakeMove()
        return best
  }
```

3　实验与结果

针对六子棋的估值设计和搜索算法，分别设计针对估值方法的对比和针对搜索方法的对比实验。并分别使用先手和后手进行测试。

针对评估函数，主要比较了基于“路”的评估方法和基于棋型的评估方法，其对比结果见表 2。

表 2　估值函数对比结果表

Tab. 2　Comparative results of valuation

对弈局数	基于路的估值胜数	胜率/%	先后手
573	521	90.92	先手
351	304	86.61	后手

由表 2 可以看出，在相同的对弈时间下，基于路的估值方法与基于棋型的估值方法相比较来说具有明显的优势。

PVS 算法与 alpha-beta 算法的对比结果见表 3。

表 3　PVS 与 alpha-beta 对弈结果表

Tab. 3　Results of PVS vs alpha-beta

对弈局数	PVS 赢的局数	胜率/%	先后手
784	779	99.36	先手
653	646	98.93	后手

PVS 算法与 UCTS 算法的对比结果见表 4。

表 4 PVS-UCTS 对弈结果表

Tab. 4 Results of PVS vs UCTS

对弈局数	PVS 赢的局数	胜率/%	先后手
637	586	91.99	先手
462	402	87.01	后手

由表3 和4 可知,与 alpha-beta 算法和 UCTS 算法相比,PVS 算法在棋力上占有一定的优势,无论是先手、还是后手,PVS 的胜率都在 85% 以上。同时,因为双方都使用相同的估值函数,从棋力方面说明 PVS 算法更适合六子棋。

4 结束语

本文针对 PVS 算法与 alpha-beta 算法和 UCTS 算法的对弈比较进行研究,结果表明 PVS 算法相较另外两种算法在公平条件下有更好的搜索效率,在限制时间的博弈比赛中有更好的优势。同时,加入置换表和哈希表、历史启发增强、迭代深化等方法提高了 PVS 算法的搜索效率,而结合基于路的估值函数则极大地提高了分支的选择效率。利用本文提出的算法开发的六子棋博弈程序获得了 2020 年辽宁省大学生计算机博弈竞赛六子棋项目冠军,进一步验证了该算法的有效性。

参考文献

[1] 邓超. 计算机围棋中的搜索算法研究[D]. 昆明:昆明理工大学,2013.

[2] WU I C, YEN S J. NCTU6 wins Connect6 tournament [J]. International Classic Guitar Association (ICGA) Journal, 2006 (3):157-158.

[3] 王静文,吴晓艺. 全国大学生计算机博弈大赛培训教程[M]. 北京:清华大学出版社,2013.

[4] 王亚杰,邱虹坤,吴燕燕,等. 计算机博弈的研究与发展[J]. 智能系统学报,2016,11(6):788-798.

[5] 徐心和,邓志立,王骄,等. 机器博弈研究面临的各种挑战[J]. 智能系统学报,2008,3(4):288-293.

[6] 何轩,洪迎伟,王开译,等. 机器博弈中搜索策略和估值函数的设计—以六子棋为例[J]. 电脑知识与技术,2019,15(34):53-54,61.

[7] 李学俊,王小龙,吴蕾,等. 六子棋中基于局部"路"扫描方式的博弈树生成算法[J]. 智能系统学报,2015,10(2):267-272.

[8] 徐长明,马宗民,徐心和. 一种新的连珠棋局面表示法及其在六子棋中的应用[J]. 东北大学学报(自然科学版),2009,30(4):514-517.

[9] YANG Xue, LI Huayu, JIANG Tianbo. Alpha-Beta-TSS in Connect6[C]//第 27 届中国控制与决策会议. 中国,青岛:《控制与决策》编辑部,2015:746-751.

(上接第 96 页)

5 结束语

采用 Netica 贝叶斯网络工具软件实现了对无人驾驶行为决策的仿真。仿真结果表明基于贝叶斯网络无人驾驶行为决策系统可以对无人车传感器收集到的各种数据进行判断,充分利用所有可能会用的信息,将定性判断与定量计算相结合描述无人车的行为决策,而且贝叶斯网络的推理功能、且辅以奖励机制来更新贝叶斯网络的结构和参数,由此推理得到的结果即能对复杂的交通环境做出更为实时、智能、安全的决策。因此应用贝叶斯网络对无人驾驶行为决策的研究必将有助于提高无人驾驶车辆在复杂场景下的决策智能性、安全性和鲁棒性,使得无人驾驶车辆在落地应用上取得了阶段性成果。

参考文献

[1] 黄璐. 基于本体论的无人驾驶车辆场景评估与行为决策方法研究[D]. 合肥:中国科学技术大学,2019.

[2] 王忠民, 曹洪江, 范琳. 一种基于卷积神经网络深度学习的人体行为识别方法[J]. 计算机科学, 2016, 43(z2):56-58,87.

[3] 蔡炳万, 石宇强, 李明辉, 等. 基于本体的贝叶斯网络知识推理研究[J]. 机械设计与制造, 2016(1):84-87.

[4] 史志富. 基于贝叶斯网络的 UCAV 编队对地攻击智能决策研究[D]. 西安:西北工业大学,2007.

[5] 熊璐, 康宇宸, 张培志, 等. 无人驾驶车辆行为决策系统研究[J]. 汽车技术, 2018(8):1-9.

[6] 谢斌. 贝叶斯网络在可靠性分析中的应用[D]. 成都:西南交通大学,2004.

[7] 俞露. 基于非同构动态贝叶斯网络的研究与应用[D]. 南京:南京大学,2017.

[8] 张琳. 基于 Ontology 和 XML 的非结构化信息语义表示机制研究[D]. 武汉:武汉科技大学,2004.

[9] 陆静, 王捷. 基于超级贝叶斯方法的专家意见先验概率修正研究[J]. 统计与决策, 2013(1):15-18.