文章编号: 2095-2163(2018) 05-0086-03 中图分类号: TP312 文献标志码: A

基于 PVS 搜索算法的亚马逊棋博弈系统的设计

李卓轩, 李 媛, 冉冠阳, 王静文
(沈阳工业大学 理学院, 沈阳 110870)

摘 要: 亚马逊棋是一种复杂度介于围棋和中国象棋之间的博弈游戏。其复杂性主要是具有极大的分支因子,在搜索过程中难以达到较高的深度。本文采用了 PVS 搜索算法,通过缩小剪枝窗口,从而有效增加剪枝效率,同时结合了历史启发增强和置换表技术,极大提高了搜索深度。使用该技术开发的亚马逊棋软件,其博弈水平得到了有效提高。

关键词: Amazons; PVS; 置换表; 历史启发

Amazons game system based on PVS search algorithm

LI Zhuoxuan, *LI Yuan*, *RAN Guanyang*, *WANG Jingwen*
(School of Science, Shenyang University of Technology, Shenyang 110870, China)

Abstract: The game of the Amazons is a game in which the complexity stands between the game of Go and Chinese Chess. Because of the huge branching factor, it is difficult to reach higher depth in the search process.Combined with heuristic and hash technology, this paper uses the PVS algorithm, and greatly improves the pruning efficiency and the search depth. The Amazons game software developed by this technology has improved the game level effectively.

Key words: Amazons; PVS; Hash; heuristic

引言

亚马逊棋是一种颇受欢迎的较新的棋盘类游戏,其行棋规则和复杂度均介于围棋和国际象棋之间。由于亚马逊棋类设计中含有较大的分支因子,使其非常适合于搜索算法的研究。亚马逊棋的棋盘构成则如图 1 所示。

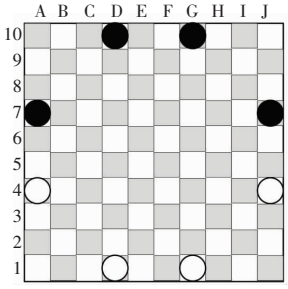

图 1 亚马逊棋棋盘

Fig. 1 Board of Amazons

研究中,将给出亚马逊棋的布棋规则可表述如下。

(1) 在 10×10 的棋盘上红方(白方) 在 A4、D1、G1 和 J4 位置上摆放白方 4 个皇后,蓝方(或黑方) 在 A7、D10、G10 和 J7 位置上摆放黑方 4 个皇后。

(2) 皇后可走棋的位置与国际象棋皇后走法的规则相同。

(3) 由红方(或白方) 开始游戏,每轮下棋由 2 步组成:

① 移动摆放皇后位置,规则和国际象棋皇后走棋的规则相同。

② 落子后以当前皇后位置为基点设置障碍,障碍摆放点的位置和皇后可摆放点的位置相同(两者使用的规则相同) 。

(4) 皇后和障碍设置的线路上不得有其它棋子或障碍。

(5) 可以完成最后一步的一方为赢家。

根据亚马逊棋的规则计算,亚马逊棋的平均分支因子达到 17 000 左右。与围棋相比,仅仅只是表现在规则相对简单,及搜索深度相对较小。因此,非常适合搜索算法的研究。

亚马逊棋的博弈系统主要由估值和搜索两大部

基金项目: 辽宁省教育科学“十三五”规划资助项目(JG16DB336) 。

作者简介: 李卓轩(1997-) ,男,本科生,主要研究方向: 计算机博弈; 李 媛(1976-) ,女,博士后,副教授,主要研究方向: 人工智能、随机过程; 冉冠阳(1996-) ,男,本科生,主要研究方向: 计算机博弈; 王静文(1965-) ,男,学士,工程师,主要研究方向: 人工智能、信息安全。

收稿日期: 2018-06-06

分组成，在本文中所用的估值研究包含 3 个方面，分别是：灵活度、位置和领域[1-2]，本文探讨的主要内容则为搜索算法，对其详述如下。

1　基于 PVS 算法的搜索引擎

研究可知，由于亚马逊棋的较高复杂度，将使其难于达到较高的搜索效率。目前，常用的方法有 UCTS 算法[3]、哈密尔顿环方法[4] 等。其中，UCTS 算法可以获得较高的搜索深度，但估值的精确性较差，而相对来说，哈密尔顿环的执行效率却会偏低。

本文采用的是基于 PVS 算法的搜索引擎，结合 Amazons 棋的特点，并且引入置换表技术和历史启发技术，该次研究旨在获得较高的搜索效率，同时能够对局面进行准确估值。

PVS 是 *alpha* - *beta* 剪枝搜索算法的一个变种算法，其设计重点在于除主变量节点外的其它所有节点都用一个零窗口(*alpha*，*beta*) 且 *alpha* = *beta* 进行搜索，遵循理念就是对浅层的节点进行整理使其基本有序，并假设第一个节点是最好的，作为主变量，展开全窗口搜索。通过零窗口搜索其它节点，判断是否存在一些节点会比当前最优值更好。如果符合 *alpha* - *beta* 剪枝则进行剪枝，假若失败则证明当初的节点不是主变量，即需对当前节点重新发起一次全窗口搜索，作为新的主变量。本文结合了历史启发增强和置换表技术，确保了搜索效率及速度。

置换表技术用于在搜索到结果的情况下记录最好的评分和方法，并在下一次搜索中直接返回相同的情况，大大提高了搜索效率。通常一个局面经搜索被判定为较好时，在其后继结点中往往有一些相似的局面也是较好的。历史启发就是建立在这样一种论点之上的。在搜索过程中，每当找到好的行棋方式时，加入一个增量来记录其历史分数，而经多次搜索均认定为是好方式的历史分数即会更高。对于即将到来的节点，可根据历史评分进行排序。如此一来，更好的行走方法(历史评分行棋方法) 就可位列在前面，从而确保搜索的效率。

在此基础上，研发推得 PVS 搜索算法的伪代码可表述如下：

```
Function PVS( node, depth, alpha, beta, gamer)
  if ( depth = 0)
  return valuation( gamer)
  else
    if ( gamer = max_gamer)
      for each child of node
        if child is first child
          value = PVS( child, depth - 1, alpha, beta, min_gamer)
        else
          value = PVS( child, depth - 1, alpha, alpha + 1, min_gamer)
        if alpha < value < beta
          value = PVS( child, depth - 1, value, beta, min_gamer)
      if ( value > alpha) alpha = value
      if ( alpha >= beta) break
      return alpha
    else
      for each child of node
        if child is first child
          value = PVS( child, depth - 1, alpha, beta, max_gamer)
        else
          value = PVS( child, depth - 1, beta - 1, beta, max_gamer)
        if alpha < value < beta
          value = PVS( child, depth - 1, alpha, value, max_gamer)
      if ( value < beta) beta = value
      if ( alpha >= beta) break
      return alpha
```

2　实验与分析

博弈系统的性能可以从胜负和访问的节点数这 2 个方面进行比较。对此可阐释分述如下。

2.1　胜负比较

胜负上的比较是对博弈系统棋力水平的直观呈现。因为每个博弈系统的最终目的便是证实自己具有较高的棋力水平。在相同限制条件下进行博弈，就可有效评判该博弈系统的水平高低。

首先双方博弈系统基于同一个估值函数，将 PVS 搜索迭代时间限定在 1 s，将 *alpha* - *beta* 剪枝搜索算法的最大深度限定在 11 层。然后设置基于 PVS 算法的博弈系统为先手，*alpha* - *beta* 剪枝算法为后手，对弈一定局数以后，先、后手互换。接着为 PVS 算法的博弈系统设置一个随机的开局，开始双方搏杀对弈，这些随机的开局将会为基于 PVS 算法的博弈系统造成一些难度。实验运行后，可以得到 PVS 算法的博弈系统胜率可参见表 1。

表 1 *alpha-beta*—PVS 对局结果表

Tab. 1 Result of *alpha-beta*—PVS

对弈局数	PVS 算法胜局数	胜率/%	先后手
610	606	99.34	先手
498	492	98.79	后手

由表 1 可知,基于 PVS 算法的博弈系统在棋力上远胜于基于 *alpha - beta* 算法的效果表现。双方基于相同的估值算法,从棋力角度说明 PVS 算法更适用于亚马逊棋。

2.2 节点比较

访问节点数上的比较反映了搜索算法在时间上的消耗,在公平的限制条件下,也折射出搜索算法在这个博弈游戏中的优劣。

本文通过 6 次函数拟合了访问节点_行棋回合函数。通过对图表的处理分析,在公平的博弈条件限制下,PVS 算法访问的节点数远超过了 *alpha - beta* 算法的最终统计数值。PVS 算法的访问节点_行棋回合函数则如图 2 所示。

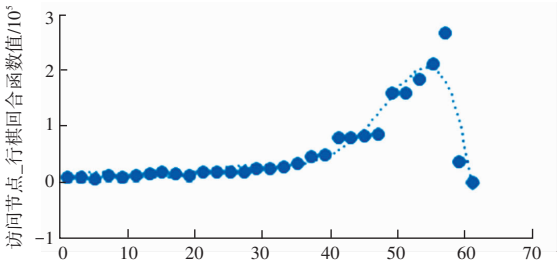

图 2 访问节点-行棋访问节点数

Fig. 2 Nodes visited-nodes visited of the game process

由图 2 中拟合的函数曲线可知,PVS 算法所访问的节点在进入残局阶段前是随行棋过程逐渐增加,直至终局阶段访问节点的显著提速下降。

alpha - beta 算法的访问节点_行棋回合函数则如图 3 所示。

由图 3 中拟合的函数曲线可知, *alpha - beta* 算法所访问的节点在开局阶段是随行棋过程逐渐增加,行棋到中局阶段访问的节点数开始下降,并一直持续至博弈终止。

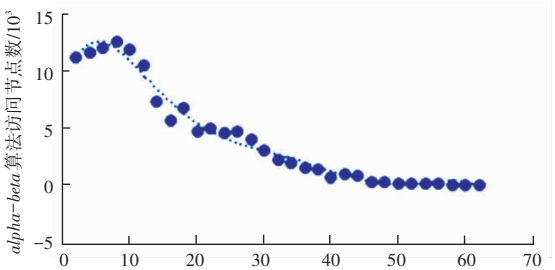

图 3 *alpha-beta* 算法访问节点数

Fig. 3 Nodes visited by *alpha-beta* algorithm

通过研究后对比可知,PVS 访问的叶子节点数远多于 *alpha - beta* 算法。究其原因即在于 *alpha - beta* 算法产生了很多剪枝,搜索的叶子节点远远少于整棵树的叶子节点,对于亚马逊棋这类走法过多、尤其开局阶段会有 1 000 多种走法的博弈游戏来说,并不适用。故而,在此方面,PVS 算法明显占优。

3 结束语

本文通过 1 000 多轮的对弈比较,利用 6 次函数去拟合访问节点_行棋回合函数,随即又设计做出了拟合后的函数图像,通过图像比较了 2 种搜索算法在这个博弈游戏中的优劣。最终结果表明,PVS 算法在开局阶段与 *alpha - beta* 算法相比并未占优领先,但是随着棋局的深入,算法优势逐渐突出,在终局阶段其优势则更加明显,由此研发获得的亚马逊棋博弈系统也将具有较高水平。

参考文献

[1] 郭琴琴,李淑琴,包华. 亚马逊棋机器博弈系统中评估函数的研究[J]. 计算机工程与应用,2012,48(34):50-54,87.
[2] LIEBERUM J. An evaluation function for the game of Amazons [J]. Theoretical Computer Science, 2005,349(2):230-244.
[3] KLOETZER J. Monte-Carlo opening books for Amazons [C]// International Conference on Computers and Games. Berlin/Heidelberg: Springer-Verlag, 2011: 124-135.
[4] BURO M. Michael Buro. Simple Amazons endgames and their connection to Hamilton circuits in cubic subgrid Graphs [C]// International Conference on Computers and Games. Berlin/Heidelberg: Springer-Verlag, 2000: 250-261.

(上接第 85 页)

参考文献

[1] 杨鹤. Web Services 在数字化校园信息集成中的应用研究[D]. 乌鲁木齐: 新疆师范大学,2009.
[2] 邵丽萍,肖世德. 新一代 Web 开发技术 ASP.NET 的发展与探析[J]. 微计算机信息,2005,21(1):190-192.
[3] 陶强. Web Service 中 UDDI 的研究与实现[D]. 武汉: 武汉理工大学,2005.
[4] 郑金芳. 基于 WEB 技术的教务管理系统的设计与实现[D]. 郑州: 郑州大学,2010 .
[5] 刘福涛. 基于 Web 的教务管理信息系统的设计与实现[D]. 大连: 大连海事大学, 2012.
[6] 王晓艳. 基于 SOA 工作流引擎的研究与实现[D]. 北京: 北京工业大学,2009.
[7] 张前峰. 基于 Web Services 的数字化校园的研究与设计[D]. 泉州: 华侨大学,2005.
[8] 王畅. 一种 SOA 的工作流管理系统的框架设计[J]. 智能计算机与应用,2018,8(1):79-81,86.
[9] 程忠岗. 基于 SOA 的工作流的研究[D]. 徐州: 中国矿业大学,2015.
[10] 王畅. SOA 架构的工作流管理系统的应用研究[J]. 绥化学院学报,2018,38(2):144-147.