ISSN 1009-3044
Computer Knowledge and Technology 电脑知识与技术
Vol.12, No.33, November 2016
E-mail: xsjl@dnzs.net.cn
http://www.dnzs.net.cn
Tel:+86-551-65690963 65690964

基于计算机博弈的五子棋AI设计

郑培铭,何丽

(北方工业大学,北京 100144)

摘要:介绍计算机博弈基础算法及部分优化算法,根据五子棋的规则,提出了针对五子棋的数据结构和算法设计,尝试将计算机博弈算法应用于五子棋,设计并实现了一个五子棋AI,并在国际赛事中取得良好成绩;实验证实使用置换表与PVS(Principal Variation Search)算法后搜索效率显著提高。

关键词:五子棋;人工智能;博弈树算法

中图分类号:TP181 文献标识码:A 文章编号:1009-3044(2016)33-0080-02

Design of Gomoku AI Based on Machine Game

ZHENG Pei-ming, HE Li

(North China University of Technology, Beijing 100144, China)

Abstract: This paper introduces the basic algorithm of machine game and some optimization algorithms. According to the rules of gomoku, the paper presents the data structure and algorithm for Gomoku. It tries to apply game tree algorithm to Gomoku, designs and implements a Gomoku AI, and achieves good results in international competitions. ; The experiment proves that the search efficiency is significantly improved by using the substitution table and PVS (Principal Variation Search) algorithm.

Key words: Gomoku; artificial intelligence; game tree algorithm

1 概述

人工智能是一门正在迅速发展的新兴的综合性很强的学科。它与生物工程、空间技术一起被并列为当今三大尖端技术。人工智能的中心任务是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。人工智能有诸多领域,博弈就是其中一种。

博弈的参加者可以是个人、集体、一类生物或机器。他们都力图用自己的智力去击败对手,博弈为人工智能提供了一个很好的试验场所。人工智能中许多概念和方法都是从博弈程序中提炼出来,并在新的技术中获得应用。许多研究成果已用于军事指挥和经济决策系统之中。

博弈问题中,最经典的就是棋类博弈。

在此以五子棋为研究样例,基于针对五子棋设计的数据结构与算法,实现了经典的计算机博弈算法。旨在证实计算机博弈算法的普适性,证明针对五子棋设计数据结构与算法的高效。

2 关键技术分析

2.1 博弈树搜索算法

任何棋类游戏都要定义一棵由博弈状态为节点的树(即“博弈树”),一个结点就代表棋类的一个局面,子结点就是这个局面进行一次状态转移可以到达的局面。根节点由决策方开始进行状态转移。博弈树的叶子节点为终结状态,即平局或者一方获胜。许多博弈问题可以使用博弈树搜索算法解决。处于树底层的结点称为叶结点(leaf node),叶结点的祖先称为内部结点(interior node)。

一个问题空间(problem space)是一个状态(state)和实现状态之间映射的操作(state)的集合。在博弈问题中,博弈树上的一个内部结点或叶结点就是一个状态,一般称为位置(position)。状态转移(move)是将一个位置转化为其子位置(successor position)的操作。如果一个位置是博弈树的叶结点,可以用估值函数(evaluator)来对其优劣进行评分(evaluate)。根据估值函数,博弈树中的每个叶结点都有一对应值(value)。博弈树搜索的目的就是找出博弈树的值(game tree value)。博弈树的值(下面简称博弈值)指的是博弈树中一个子结点的值,该值对于博弈双方都是最优的。博弈树的子树在该子树搜索完成之后也会返回一个博弈值,该值是子树的局部最优值。

计算一棵博弈树的博弈值,可以使用基本的极大极小树搜索(Min-Max Tree Search)。为了减少搜索的节点数,可以使用Alpha-Beta算法进行剪枝。在Alpha-Beta算法的基础上,各种改进方法被先后提出来,例如置换表,PVS算法以及并行Alpha-Beta算法。在博弈树搜索算法方面,前人做了许多丰富而充满意义的研究工作。

收稿日期:2016-10-20
基金项目:2016年北方工业大学大学生科技活动项目,2015年优秀青年教师培养计划项目(XN132)

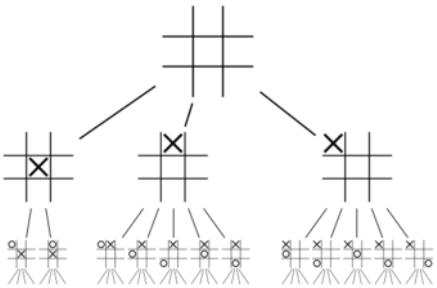

图1　井字棋的博弈树

2.1 针对Alpha-Beta算法的改进

在Alpha-Beta算法被广泛运用后,对该算法的很多改进算法也相继被提出.这些改进算法主要在以下几个方面对Alpha-Beta算法进行改进:

1)择序。在搜索博弈树时,内部结点有多个可能的状态转移。择序指的是搜索这些分支的顺序。

在Alpha-Beta算法中,提高择序的好坏就意味着提高剪枝发生的概率。所以一个好的状态转移可以定义为:

①导致剪枝的移动。

②或者如果没有导致剪枝,但是产生了最小最大值。

一个好的状态转移并不一定是最优的状态转移(best move),因为一个导致剪枝的状态转移可能引起了该子结点上的搜索停止,但是并不意味着其后的子结点的搜索不会产生一个更好的值。

2)搜索窗口的大小。搜索窗口的大小指的是β和α之差.搜索窗口越小,发生剪枝的概率越大。

3)信息的重用。保存子树的搜索结果,并审查这棵子树是否在随后的搜索中再次出现。如果该子树再次出现,那么关于子树的搜索结果的信息就可重复使用。

常见的改进算法有迭代加深算法、PVS(Principal Variation Search)算法与置换表等。实验证实,使用置换表与迭代加深算法提供启发性的估值,并用于状态转移的排序后,良有序的状态转移使PVS有更高的效率。使得搜索效率大幅提高。

3 五子棋数据结构与算法设计

3.1 棋盘数据结构

由于在PC上棋盘的访问不是性能瓶颈,所以我们用可读性较强的方法来定义显式的棋盘,通过调用指定接口落子,我们还能维护一个编码后的棋盘。

1)显式的棋盘

定义一个Board[SIZE][SIZE]数组。其中,SIZE是指可落子的棋盘范围+8。因为在判断棋型的过程中,需要访问相邻4格内的点(9格棋型:4 + 1 + 4 = 9),为了便于操作与编码,在棋盘围增加4格的预留空间。

2)编码后的棋盘

首先我们分别定义二进制数$[00]_2$,$[01]_2$,$[10]_2$为空、黑、白状态,$[11]_2$为预留空间。再定义四个数组L[POS],R[POS],R[POS],W[POS],其中保存着在四个方向(横竖撇捺)上棋盘的二进制编码。例如11111111000001111111就是5*5的空棋盘在第一行上的编码。也就是说,在二进制棋盘中,每一个格子占两个比特。

3.2 着法生成

着法生成是指,一个局面(状态)下有几个可能的走法(状态转移)。最简单的着法生成是将所有空点纳入着法序列。但实际上,有价值的走法一般在已落子的点相邻3格内。所以只需维护一个表C[SIZE][SIZE],其中C[i][j]保存着点[i, j]三个内的棋子数。这些信息同样可以用于择序。

3.3 着法择序

使用估值函数Evaluate()得到值v。

择序的伪代码

```
sort_moves()
1:    for i < 1 to node.moves.length
2:    node.moves[i] = Evaluate(node.moves[i])
3:    sort(node.moves)
```

3.4 棋型与棋型表

各种棋型的定义。一个棋型只能有一个类型。优先级从高到低。

1.成五:连续的五个及以上的同色棋子的棋型。
2.活四:有两个落同样颜色子后能成五的点的棋型。
3.冲四:只有一个点落同样颜色子后能成五的棋型。
4.活三:有落子后能成活四的点的棋型。
5.眠三:有落子后能成冲四的点的棋型。
6.活二:有落子后能成活三的点的棋型。
7.眠二:有落子后能成眠三的点的棋型。
8.活一:有落子后能成活二的点的棋型。
9.眠一:有落子后能成眠二的点的棋型。
10.无效:双方都不可能成五的棋型。

根据这些规则,可以递归的生成可查询的棋型表Pattern。

显然,棋型表的大小只与棋型长度有关,一个长9格的棋型对应18位的编码。可求出棋型表的大小为2^{18} byte。即使棋型拓展到11格,2^{22} byte也是可以接受的大小。

3.5 估值设计

定义向量V (v1, v2, v3 ... , v10) 分别对应十种棋型的分值(value)。

定义向量C (c1, c2, c3 ... , c10) 分别对应棋盘上十种棋型的数量。

通过手动设置或者调参算法给定V以后,在搜索过程中维护C。到达叶节点时调用估值函数。

估值函数执行简单的向量运算。

向量运算可以使用SIMD进行优化,从而得到更高的效率。

3.5 实验数据及结果

利用本文设计的五子棋数据结构和对博弈算法的改进;实现了一个五子棋AI,在国际赛事中取得良好成绩。结果如下所示。

其中Chis为本文实现的AI。Gomocup 1的结果为决赛,Gomocup 2为半决赛。Chis在freestyle中获得第13名。

(下转第90页)