面向目标的博弈搜索策略及其应用

张 越[1],芦东昕[1,2]

(1.华北电力大学 计算机科学与技术系,北京 102206;
2.中兴软件技术有限公司成都研究所,四川 成都 610041)

摘 要:博弈是人工智能研究的重要分支,它涉及人工智能中的推理技术、搜索方法和决策规划。而搜索策略是博弈问题的关键。针对搜索技术中存在的由于搜索空间巨大而引起的搜索效率下降的缺点,结合五子棋的特点,探讨了相应博弈问题的求解策略,提出一种结合PVS算法、静态着法启发、历史启发算法的搜索策略。实验结果证明,该算法不但能保证博弈水平,还能得到较好的搜索效率。

关键词:博弈;负极大值搜索;极小窗口搜索;历史启发;静态着法启发

中图分类号:TP18;O225 **文献标识码**:A **文章编号**:1673-629X(2007)03-0102-04

Goal-Oriented Search Strategy for Game and Its Application

ZHANG Yue[1], LU Dong-xin[1,2]

(1. School of Computer Science & Technology, North China Electric Power University, Beijing 102206, China;
2. Chengdu Institute, Zhongxing Software Technology Co. Ltd, Chengdu 610041, China)

Abstract: Game is one of the major research realms of artificial intelligence. It involves reasoning, search and planning. The search efficiency will descend when the problem state is too large. Thus, this paper explores a strategy for solving the corresponding game problem, proposing a search strategy which combines PVS, history heuristic and rule heuristic based on the gobang. The result shows that the algorithm can not only guarantee the game level, but also improve its search efficiency.

Key words: game; negamax; PVS; history heuristic; rule heuristic

1 人工智能与博弈问题

人工智能是一门正在迅速发展的新兴的综合性很强的边缘科学,它与生物工程、空间技术一起被并列为当今三大尖端技术,它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。

博弈,就是对策或斗智,它涉及人工智能中的推理技术、搜索方法和决策规划。作为人工智能研究的重要分支,数十年来,受到了广泛的关注,人们为之投入了大量的研究。

2 基本搜索策略

对于计算机来说,直接通过当前的棋盘信息来判别走法的好坏并不精确。除了输赢这样的局面能可靠地判别以外,其他的判断都只能做到大致估计。判断两种走法孰优孰劣的一个方法是向下搜索若干步,比较棋局发展下去的结果。因此搜索策略就成为博弈问题的关键,策略的优劣直接影响程序的效率、智能的高低。

目前绝大部分博弈算法都是基于冯·诺依曼(John Von Neumann)于1928年提出的极大极小博弈树理论。1963年Brudno成功将这一理论发展,根据避免搜索极大极小博弈树中的不必要的节点,形成了α~β剪枝搜索[1]。而PVS算法正是在α~β算法的基础上,根据一种最优解假设思想产生的算法。麻省理工学院的国际象棋机器人StarSocrates就是采用的PVS算法。

2.1 博弈树

博弈树就是指用来描述博弈过程的与/或树,是从根部向下递归产生的一棵包含所有可能的对弈过程的搜索树[2],如图1所示。它有以下特点:

(1)树的根部是棋局的初始局面。

(2)“或”节点和“与”节点逐层交替出现,且轮流地

收稿日期:2006-06-12

基金项目:中国下一代互联网示范工程(CNGI)移动奥运项目资助(CNGI-04-17-2A)

作者简介:张 越(1982-),女,河南人,硕士研究生,研究方向为嵌入式操作系统、人工智能;芦东昕,博士后,教授,研究方向为第3代移动通讯、人工智能。

扩展节点,每个节点对应每一种可能走法所生成的局面。

(3)树的末梢是结束的棋局,一方获胜或双方平局。

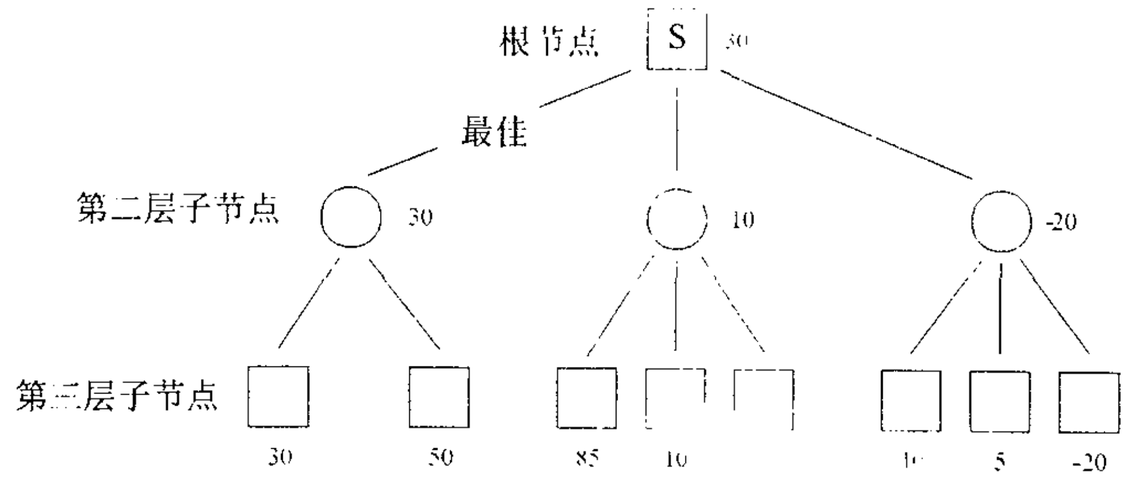

图 1　博弈树示意图

在大多数博弈问题中,要想建立完全搜索树是不可能实现的。以五子棋为例,按照 Victor Allis 的分析:以每一局面平均 40 种走法计算,建立一棵(双方各走 50 步)搜索树需要生成 10^{160}个节点[1]。因此这种完全取胜策略必须丢弃,而把目标定为从局部搜索树中选取一步最好的走步,等对方走步后再寻找下一步好棋,可通过负极大值搜索方法来达到这种搜索策略。

2.2　负极大值搜索算法(Negamax Algorithm)

Knuth 和 Moore 于 1975 年提出了负极大值方法[3],避免了每次进行极大或者极小的比较之前,都要检查取极大值还是极小值,实现了对极大极小算法的改进。负极大值算法的核心是:父节点的值是各子节点值的负数的极大值。

假设甲、乙两个玩家对弈,甲先走,之后两人交替走步直到游戏结束。负极大值搜索过程的步骤如下:

(1)由于不可能对整棵博弈树进行搜索,所以建立一棵固定深度的搜索树,其叶子节点不必是终了状态,而只是固定深度的最深一层的节点。

(2)在一棵博弈树中,假定令甲胜的局面值为 10000,乙胜的局面值为 -10000,和局的值为 0,其他情形下依据双方棋子的棋形评定为 -10000～10000 之间的具体分数。用静态估值函数 f 对每个叶节点进行估值,对一个该甲方走棋的局面返回正的估值,对于一个该乙方走棋的局面返回负的估值。

(3)计算上层节点的倒推值。取其子节点负数的最大值。

(4)对弈双方,都选择子节点值最大的走法。

负极大值搜法算法的时间复杂度是 $o(b^n)$,这里 b 是分枝因子,n 是搜索的最大深度[3]。对于分枝因子在 40 左右的棋类游戏,时间开销随着 n 的增大会急剧地增长,不出几层就会超出计算机的处理能力。

人们在开发高效的搜索算法上投入了大量的研究,改进搜索算法的目标在于将不必搜索的(冗余)分枝从搜索的过程中尽量剔除,以达到搜索尽量少的分枝来降低运算量的目的。在过去的几十年中,一些相当成功的改进大大提高了极大极小搜索的效率。$\alpha \sim \beta$ 剪枝、迭代深化、置换表、历史启发等手段的综合应用将搜索效率提高了几个数量级。

3　算法设计

搜索算法是棋类对弈程序的核心算法,在众多搜索算法中,如何选择适合自己的算法,并有效地把它们组合起来,是决定搜索效率的关键所在。

文中结合五子棋的特点,提出一种结合 PVS 算法、静态着法启发、历史启发算法的搜索策略,得到了较好的搜索效率。

3.1　五子棋形式描述

五于棋以某方在棋盘某直线位置上有相连的同色五子作为胜负标准。现定义一直线上的如下几类落子姿式:

a2:表示活二,即两个同色子相连且两端无异色子阻挡;

a3:表示活三,即三个同色子相连且两端无异色子阻挡;

a4:表示活四,即四个同色子相连且两端无异色子阻挡;

b2:表示有两个同色子,中间隔一空位置,且两端无异色子阻挡;

b3:表示活嵌四,即有三个同色子,其中有两个相连,另一个与相连两子隔一空位置,且两端无异色子阻挡;

b4:表示活嵌五,即相连五个位置上有四个同色子,它们不全相连,其中有一个空位置;

c3:表示 a3 或 b3 有一端被异色子阻挡;

c4:表示 a4 或 b4 有一端被异色子阻挡;

aim:表示目标姿式,即至少有五个同色子相连。

3.2　极小窗口搜索(PVS 算法)

A. Reinefeld 于 1982 年提出了该算法。此算法基于如下假设[2]:假定第一步是最佳的移动,其后继则次之,直到另一节点被证明是最佳的。在一个中间节点,其第一个分枝以一个完整窗 口(α,β) 搜索之并产生一个位于该窗中的值 v,后继的分支则以一个极小的窗口$(v,v+1)$搜索之。极小窗口搜索的意图在于使用极小的窗口建立极小的搜索树,以此达成高效的搜索。本质上讲,极小窗口搜索依赖于后继节点的值相对于前驱节点值的微小变化的猜测,如果猜测被证实不准确(fail high),随后就需以$(v+1,\beta)$为窗口重新搜索。

由于以$(v, v+1)$为窗口的搜索剪枝效率远高于更大的窗口，所以极小窗口搜索的效率很高，由于 fail high 以后重新搜索相对于完整窗口缩小了范围，其搜索效率也比以完整窗口进行的搜索效率高，而且经过历史启发排序后的第一个节点有些情况下就是最优节点，所以使用该算法效率较高。

PVS算法的伪代码如下：

```
int PrincipalVariation(depth,alpha,beta)
{
  if(GameOver)
    return Evaluate(pos);//胜负已分,返回估值
  if(depth == 0)
    return Evaluate(pos);//叶节点,返回估值
  make move m; //创建第一个子节点
  //用全窗口搜索第一个节点
  best = -PrincipalVariation(depth-1,-beta,-alpha);
  unmake move m;//撤销第一个节点
  for(each possible move m) //从第二个节点开始,对每一个节点执行以下操作
  {
    if(best<beta)
    {
      if(best>alpha)
        alpha = best;
      make move m;//创建子节点
      //空窗探测
      value = -PrincipalVariation(depth - 1,-alpha-1,-alpha);
      //如果探测结果 fail-high,重新搜索
      if(value >alpha&&value<beta)
        best = -PrincipalVariation(depth - 1,-beta,-value);
      else if(value > best)
        best = value; //命中
      unmake move m;//撤销子节点
    }
  }
  return best;//返回最佳值
}
```

3.3 静态着法启发

静态着法启发是指不依赖于搜索结果的着法排序方式，即程序分析了某个局面后，不经过搜索，就大致应该知道哪些着法应该首先尝试[4]。

具体对于五子棋来说，相连的同色子越接近五个，则越接近胜利，显然如果某方出现有活四或双活三等情况时必胜无疑，这时无需再按 PVS 算法进行搜索。

确定最佳优先走步位置的算法描述如下(假定对弈双方分别为甲、乙)：

步骤 1：判别当前棋局中是否存在非常接近目标姿式的特殊姿式(如 b4 与 a3 等)。若存在，则确定落子位置，这时构成甲方姿式 a5 或 b4，或者破坏乙方姿式 b4 或 c4 等，转向步骤 5；否则执行以下步骤。

步骤 2：判别当前棋局中是否存在其它姿式。当不存在时转向步骤 3，否则对允许落子范围内相应节点有最高优先级的一切落子候选位置执行以下步骤。

步骤 2.1：若某位置处落子后生成的新棋局的姿式(组合)表明必胜无疑(如多次出现 b4 或 c4，或者同时出现 b4 与 c4)，则确定此位置落子，转向步骤 5；否则执行以下步骤。

步骤 2.2：把当前棋局相应的节点作为当前叶节点，且让 a 为比一切可能的倒推值更小的最小值。

步骤 2.3：按落子后生成的新棋局的姿式(组合)变化及离达到目标姿式的程度，对节点(位置)赋以优先级，离达到目标姿式的程度相近者具有相同的优先级。

步骤 2.4：对粗略好处估值表明谁落子便对谁有利的一切位置的相应节点计算落 MAX 方之子时的倒推值，以超过 a 的倒推值作为 a 值，最终 a 值为最大倒推值，以相应于最大倒推值的节点所对应的位置为落子位置，转向步骤 5。

步骤 3：在允许落子范围内确定具有最大静态估值的落子位置，这时对于甲方 a2 出现次数最多，转向步骤 5。当不能确定这样的位置时，执行步骤 4。

步骤 4：以乙方落子处的相邻位置为落子位置。

步骤 5：结束。

节点的排列顺序及搜索的节点数量对搜索的速度起着至关重要的作用。考察上述算法：

(1)节点排序。

步骤 2 对允许落子范围内各个落子的候选位置，依据落子在此处时呈现的姿式(组合)的变化及离达到目标姿式的程度赋予相应节点以不同的优先级。仅具有最高优先级的那些节点才可能进行倒推值计算。此时进一步仅对落子时粗略好处估值表明谁落子就对谁有利的那些位置的相应节点进行倒推值计算(对某未落子位置的相应节点的粗略好处估值，按照在此位置落子后呈现的姿式及姿式的变化情况而作出。当有利于达到或接近目标姿式时，粗略好处估值则大)。这样便在进行倒推值计算之前对节点进行了排序，仅生成具有最高优先级的相应节点。从而提高了搜索效率。

(2)减少搜索的节点数目。

五子棋棋盘可有 361 个落子位置，但通常情况下，不论哪一方必将在有子处邻近落子。可以产生一个棋面限制框，记录当前棋面所有棋子的最左最右最上最

下点构成的矩形,而且认为下一步棋的位置不会脱离这个框 3 步以上。另外,考虑到已落子位置不能再落子,以及当未落子的某位置其周围两行两列两对角线共 16 个位置处有落子,该位置也不作为落子候选位置,这样就大大减少了候选节点的数量。步骤 3、4 中,采用该种方法缩小落子范围与限制落子位置,进一步提高了搜索的效率。

由此可见,上述算法基于离达到目标姿式的程度,对节点进行按优先级排序,仅对优先级最高的那些节点才进行倒推值计算,从而提高了搜索效率。

3.4　历史启发算法(History Heuristic)

对于博弈树来说,节点的排列顺序是杂乱无章的。严格来说,如果搜索树在每个节点的分枝因子都是 b,那么 d 层 $\alpha \sim \beta$ 搜索在最好情况下搜索的节点数 n 如公式 1 所示[5]:

$$n = b^{(d/2)} + b^{[(d+1)/2]} - 1 \qquad (1)$$

根据 Knuth 和 Moore 所作的研究表明,使用 $\alpha \sim \beta$ 搜索建立的博弈树的节点个数在 n 和使用极大极小搜索建立的节点数之间。由此可见,$\alpha \sim \beta$ 剪枝与节点的排列顺序高度相关,因此,如何调整待展开的走法排列的顺序,是提高搜索效率的关键。

J.Schaeffer 提出了历史启发[1] 的方法。狭义上所说的启发,即通过排序尽量首先搜索最好的着法。

在基于 $\alpha \sim \beta$ 的搜索当中,一个好的走法可以定义如下:

(1) 由其产生的节点引发了剪枝。

(2) 未引发剪枝,但是其兄弟走法中的最佳者。

"历史启发"就在搜索的过程中,每当找到一个好的走法,就将与该走法相对应的历史得分作一个增量,使其具有更高的优先被搜索的权利。当搜索中间节点时,将走法根据其历史得分排列顺序,以获得较佳的排列顺序。

文中采用的算法首先利用上面提到的静态着法进行排序,然后再结合历史启发算法进行排序,进一步提高了搜索效率。

3.5　实验结果

实验证明,使用文中提出的搜索算法,搜索效率有了一定的提高。实验数据如表 1 所示。

4　算法可能的改进

对于上述算法,当搜索超过 5 层时,便会变慢,下一步甚至要几分钟。对上述算法还可以进行以下一些改进。

表 1　各种搜索方法在开局时每走一步的平均耗时/ms

搜索深度 \ 搜索算法	α~β	PVS	PVS+历史启发
2	10	10	4
3	215	160	40
4	2000	1370	195
5	41954	16459	2905

注:测试所用的硬件环境为 P4 2.8GHz,内存 512M。

4.1　知识系统

可以加入知识系统,即把以前的精彩棋局加入到知识结构中,在搜索过程中分析棋盘上的棋形,再根据数据库中相应的棋局,依棋局下棋。

4.2　学习系统

上述算法没有自学习的能力,这样 AI 在下棋时会重复以前出现过的"昏招"。因此可以记录某一走法(一盘棋的最后几步)导致输赢的概率,在每盘棋结束时,对概率进行更新。下次对弈时,AI 如果发现当前布局的这种走法在以前输过棋,那么它不会选择这个走法,从而达到自学习的效果[6]。

5　结　论

搜索算法是棋类对弈程序的核心算法,是决定搜索效率的关键所在。文中结合五子棋的特点,定义了落子姿式的概念,并根据姿式,提出一种结合 PVS 算法、静态着法启发、历史启发算法的搜索策略,同时提出了可以通过加入知识系统、增加机器学习的方法对算法进行改进。实践证明,文中提出的算法效果较好,可供其它博弈问题求解参考。

参考文献:

[1] 王小春.游戏编程(人机博弈)[M].重庆:重庆大学出版社,2002.

[2] 蔡自兴.人工智能及其应用[M].北京:清华大学出版社,1999.

[3] 陆汝铃.人工智能[M].北京:科学出版社,1999.

[4] Pear J. Heuristics, Intelligent Search Strategies for Computer Problem Solving[M]. [s. l.]: Addison - Wesley Publishing Company, 1984.

[5] 蒋加伏,陈蜀祥,唐贤英.基于知识推理的博弈树搜索算法[J].计算机工程与应用,2004(1):74 - 77.

[6] 肖齐英,王正志.博弈树搜索与静态估值函数[J].计算机应用研究,1997(4):74 - 76.