# 机器博弈及其搜索算法的研究

张 振[1], 庞 海[2]

(1.武汉理工大学 计算机学院, 湖北 武汉 430063; 2.郑州大学 升达经贸管理学院, 河南 郑州 451191)

摘 要: 机器博弈是人工智能一个传统的研究领域。从机器博弈的基本理论出发, 介绍了机器博弈理论和机器博弈系统的一般构成, 重点阐述了现今已存在的各种机器博弈搜索算法及其优缺点。

关键词: 博弈系统 ; 博弈搜索算法; 极大极小值算法; Alpha- beta剪枝算法

中图分类号: TP312 文献标识码: A 文章编号: 1672- 7800(2008)07- 0048- 03

## 0 引言

博弈一向被认为是富有挑战性的智力游戏, 有着难以言状的魅力。博弈虽然自古就是人与人的对弈, 但自从有了计算机以后, 人们就有了用计算机下棋的想法, 早在上世纪50年代, 就出现了若干博弈程序。最近的一次是1999年IBM的 '深蓝 '战胜了国际象棋世界冠军卡斯帕罗夫, 惊动了世界。机器博弈的核心技术是博弈搜索算法, 它与估值及规则(走法)构成一个完整的系统。

## 1 机器博弈的基本思想

机器博弈的核心思想并不复杂, 实际上就是对博弈树节点的估值过程和对博弈树搜索过程的结合。在博弈的任何一个中间阶段, 站在博弈双方其中一方的立场上, 可以构想一个博弈树。这个博弈树的根节点是当前时刻的棋局, 它的儿子节点是假设再行棋一步以后的各种棋局, 孙子节点是从儿子节点的棋局再行棋一步的各种棋局, 以此类推, 构造整棵博弈树, 直到可以分出胜负的棋局。整棵的博弈树非常庞大, 且不同的棋类有所不同, 分支因子大的如围棋的博弈树显然要比分支因子小的如国际象棋的博弈树要大得多。博弈程序的任务就是对博弈树进行搜索找出当前最优的一步行棋。对博弈树进行极大极小搜索, 可以达到这一目的。极大极小搜索, 是因为博弈双方所要达到的目的相反, 一方要寻找的利益恰是一方失去的利益, 所以博弈的一方总是希望下一走步是儿子节点中取值最大者, 而另一方恰恰相反。这便形成了极大极小过程。当然, 程序不能也没有必要做到搜索整棵博弈树的所有节点, 对于一些已经确定为不佳的走步可以将根节点的子树剪掉。而且, 搜索也不必真地进行到分出胜负的棋局, 只需要在一定深度范围内对局面进行评价即可。只有搜索空间缩小到一定程度, 搜索才可以真正地进行。当搜索进行到一定深度, 用局面评价机制来评价棋局, 按照极大极小的原则选出最优, 向上回溯, 给出这一局面的父亲节点的价值评价, 然后再继续向上回溯, 一直到根节点, 最优走步就是这样搜索出来的。在这个过程中, 最为重要的是搜索算法, 高效的搜索算法可以保证用尽量少的时间和空间损耗来达到寻找高价值的走步。但是真的想要博弈程序棋力提高, 还必须有一个好的局面评价机制, 即估值算法作后盾。就是说, 用这个估值算法评价的局面价值必须是客观的、正确的, 可以确凿的评价局面的优劣以及优劣的程度。

## 2 机器博弈系统

根据机器博弈的基本思想, 可以确定一个机器博弈系统的一般构成。首先是知识表示的问题, 选用一种合适的方法记录棋局。这时需要考虑在这种知识表示的数据结构之上将要进行的各种操作, 知识表示最经常进行的操作所花费的时间和空间代价应该最小。其次, 根据不同的棋类的不同规则集, 要有一个相应的走法产生机制。它的作用是用来产生整棵博弈树, 即处于博弈树的任何一个节点位置上, 应该能够运用该机制产生这个节点的所有儿子节点, 也就是接下来的所有合法走步。除了以上两个模块以外, 就是博弈核心的搜索技术和与之配合的估值技术了。这4个部分相互配合运转起来, 就可以实现机器博弈。

## 3 基本搜索算法

### 3.1 极大极小值算法

假设有2个博弈者甲和乙(甲为计算机, 乙为棋手), 在这个博弈事件中我们要为甲找到最佳的走步。现假设甲方先下, 乙

作者简介: 张振(1982~), 男, 河南禹州人, 武汉理工大学计算机学院硕士研究生, 研究方向为人工智能、计算机网络; 庞海(1981~), 男, 河南开封人, 郑州大学升达经贸管理学院助教, 研究方向为计算机应用。

方后下,深度为奇数的节点为甲方下了一子之后的局面(即乙方选择下法的局面),称之为极小节点;深度为偶数的节点为乙方下一子之后的局面(即甲方选择下法的局面),称为极大节点;博弈树的顶节点深度为0。一个最佳走步可以由一个极大极小化过程产生,甲方要从搜索树中选择拥有最大值的节点,因此,一个标有'甲'的节点的估计值可以由它的标有'乙'的子节点的最大值确定。另一方面,乙方从叶节点中选择时,由于估计值越小对它越有利,因此必然选取估计值最小的节点(即最负的估计值),因此,标有'乙'的节点估计值可由它的标有'甲'的子节点的最小值确定。综上所述,可由博弈树的叶节点出发向上一层层倒推,得到上一层的估计值,从而得出根节点的估计值,这样就可以确定从根节点出发的最佳走步,称之为极大极小值算法,其伪代码如下:

```
double ji- da- ji- xiao(int depth)
{
int i;
double best, n;
if (比赛结束)
return evaluation ();
if (depth= =0)
return evaluation (); /*叶节点*/
if (甲方)
best = - 1 000;
else
best =1 000; /*初始化当前最优值,假设1000为一个不可能达到的值*/
for (i=1;i<=w;i++)
{
n= ji- da- ji- xiao (depth- 1);
if(n>best&&甲方)
    best=n;
if(n<best&&棋手)
    best=n;
}
return best;
}
```

### 3.2 Alpha- beta剪枝算法

在极大极小搜索的过程中,存在着一定程度的数据冗余,剔除这些冗余数据,是减小搜索空间的必然做法,所采用的方法就是1975年Monroe Newborn提出的Alpha- beta剪枝。把Alpha- beta剪枝应用到极大极小搜索或者负极大值搜索中就形成了alpha- beta搜索。其伪码如下:

```
double Alpha- Beta (double alpha, double beta, int depth );
{
int i;
double n, best;
if (比赛结束)
  return evaluation ();
if (depth= =0)
  return evaluation (); /*叶结点*/
if (极大节点)
{
for(i=1; i<=w; i++)
{
   n= Alpha- Beta (alpha , beta , depth- 1);
if(best>alpha)
alpha = best;
}
return alpha;
}
else /*极小节点*/
{
  for(i=1; i<=w; i++)
  {
    n= Alpha- Beta (alpha , beta , depth- 1);
    if(best<beta)
     beta= best;
  }
  return beta;
  }
}
```

## 4 Alpha- beta的增强算法

### 4.1 渴望搜索

在alpha- beta剪枝过程中,初始的搜索窗口往往是从- ∞(即初始的alpha值)到+∞(初始的beta值),在搜索进行中再不断缩小窗口,加大剪枝效果。渴望搜索就是渴望从一开始就使用小的窗口,从而在搜索初起,就可以进行大量的剪枝。这个初始窗口的选定很重要,如果选择准确,即所要寻找的走步就在这个窗口内,搜索便可以继续进行。如果第一步搜索的返回值证明最佳走步大于beta值,就需要重新确定窗口。以原来的beta值为alpha值,以+∞为新的beta值重新搜索。相反如果第一步的返回值证明最佳走步小于alpha值,重新确定的窗口就是以- ∞为alpha值,原来的alpha值为beta值了。可见第一次搜索猜测的窗口非常重要,如果猜测准确,搜索时间可以大大缩短,如果不准确,这一优势就不明显了。由于渴望搜索是一种基本的搜索方法,它在和后面叙述的遍历深化结合使用的时候,就可以借助前一次深度为depth的搜索的结果来猜测当前深度为depth+1搜索的窗口了。

### 4.2 极小窗口搜索

极小窗口搜索,也被称为是主变量导向搜索(Principal Variation Search),常常简称为PVS算法或者NegaScout算法。这个算法应用非常广泛。它的出发点是和渴望搜索大致相同的,即用极小的窗口来限制剪枝范围。它的过程是这样的:在根节

点处,假定第一个儿子节点为主变量,也就是假定它为最佳走步,对它进行完整窗口(a, b)的搜索并得到一个返回值v,对后面的儿子节点依次用极小窗口(也被称为是零窗口)(v, v+1)来进行搜索,如果搜索返回值大于零窗口,则证明这一分支亦为主变量,对它进行窗口为(v+1, b)的搜索,可是如果返回值小于零窗口,这一分支就可以忽略,因为它的最佳走步还不如已有的走步。极小窗口搜索采用了极小的窗口,剪枝效率最高,并且只对主变量进行大窗口的搜索,所以大部分搜索动作是有效的,搜索产生的博弈树也很小。

### 4.3 置换表

在搜索进行中,查询所有的走步,经常会在相同的或者不同的路径上遇到相同的棋局,这样的子树就没有必要重复搜索,把子树根节点的估值、子树的最好走步和取得这个值的深度信息保存在一个称为置换表的数据结构中,下次遇到时直接运用即可。但是,对不同的情况要区别对待。对于固定深度的搜索,如果前一次保存的子树深度小于新节点要搜索的深度,还是要进行重新的搜索以保证所取得数据的准确程度。反之,如果置换表中保存的子树信息表明,子树的深度大于或者等于当前的新节点要求的搜索深度,它的信息就可以直接运用了。置换表增强如果和有效的alpha- beta搜索结合使用,结果就会使实际搜索的博弈树小于最小树。博弈树搜索的问题实际上是一个有向图的搜索。那些在置换表中被命中的节点,就是有向图中若干路径的交叉点。只有避免这些交点被重复搜索,搜索过程中生成的搜索树才有可能小于最小树。这也是后来证明深度优先的算法并不比最佳优先的算法差的原因之一。对置换表的查找和插入操作不成为整个搜索的负担,这种置换表的构造才能提高效率。一般使用散列表来实现。

### 4.4 遍历深化

遍历深化是因对博弈树进行多次遍历,又不断加深其深度而得名,有人还把它称为蛮力搜索。算法利用了alpha- beta算法对子节点排序敏感的特点。它希望通过浅层的遍历给出节点的大致排序,把这个排序作为深层遍历的启发式信息。另外,算法用时间控制遍历次数,时间一到,搜索立即停止,这也符合人类棋手的下棋特点。在关键的开局和残局,由于分支较少,也可以进行较深层次的搜索。算法的过程是,对以当前棋局为根节点的博弈树进行深度为2的遍历,得出其儿子节点的优劣排序,接着再从根节点进行深度为3的遍历,这一次优先搜索上次遍历中得出的最优者,从而加大剪枝效果,以此类推,在进行第3次、第4次的遍历,一直达到限定时间为止。由于这个算法的每次遍历都从根节点开始,所以有人称其为蛮力搜索,但实际上由于每次都可以优先搜索相对好的节点,导致剪枝效率增大,其实算法效率是很高的,目前这一算法得到了广泛的认可。

### 4.5 历史启发搜索

历史启发也是迎合alpha- beta搜索对节点排列顺序敏感的特点来提高剪枝效率的,即对节点排序,从而优先搜索好的走法。所谓好的走法即可以引发剪枝的走法或者是其兄弟节点中最好的走法。一个这样的走法一经遇到,就给其历史得分一个相应的增量,使其具有更高的优先被搜索的权利。杀手启发实际上是历史启发的特例。它是把同一层中,引发剪枝最多的节点称为杀手,当下次搜索时,搜索到同一层次,如果杀手走步是合法的话,就优先搜索杀手。

## 5 结束语

本文讨论了博弈搜索的几种算法,其中α-β剪枝算法比较成熟,是当前最常用的算法,在融合各种策略后具有很高的剪枝效率。如果能进一步改进数据结构和进行代码优化以及使用开局、残局库,这都可以使程序具有很高的效率智能。

参考文献:

[1] 敖志刚.人工智能与专家系统导论[M].合肥:中国科技大学出版社,2002.

[2] NilsJ, Nilsson.Artincial Intelligence.A new synthesis[M].北京:机械工业出版社,2000.

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

[4] 蔡自兴.人工智能及其应用[M].北京:清华大学出版社,1999.

[5] 徐心和,王骄.中国象棋计算机博弈关键技术分析[J].小型微型计算机系统,2006(6).

(责任编辑:杜能钢)

# Research of Machine Gambling and Searching Algorithm

Abstract:Machine gambling is a traditional research field of artificial intelligence . This article ,begins with the elementary theory of the machine gambling, introduces the machine gambling theory and the general constitution of the machine gambling system ,especially elaborates all kinds of searching algorithm of machine gambling and its advantages and disadvantages.

Key Words: Gambling System; Gambling Searching Algorithm; Minimax Algorithm; Alpha- beta Pruning Algorithm