```
        jne setwA
        jmp imgalt
alt_end ret
        show_image2 endp
        pixel_alt proc near;修改颜色分量
        push cx
        xor dh,dh
alt1: mov al,fs:[esi]
        inc esi
        mov ah,es:[di]
        cmp al,ah
        je next_color
        mov succ_flag,1
        ja alt11
        sub ah,al
        add ah,increase
        cmp ah,maxcdif
        jb next_color
        dec byte ptr es:[di]
        jmp next_color
alt11: sub al,ah
        add al,increase
        cmp al,maxcdif
        jb next_color
        inc byte ptr es:[di]
next_color: inc di
        loop alt1
        cmp dh,0
        je alt2
        jmp line_fin
alt2 pop ax
        mov cx,[bp+4]
        cmp ax,cx
        je line_fin
        sub cx,ax
        set_windowA
        xor di,di
        mov dh,1
        jmp alt1
line_fin ret
pixel_alt endp
code ends
        end start
```

## 参考文献


1 侯阳.微机图形文件模式集粹。北京:学苑出版社,1993.12

2 施威铭研究室.SVGA显示原理和绘图技巧.北京:学苑出版社,1995.11

3 梁肇新.在DOS实模式下直接存取4GB内存。计算机世界月刊,1995年第12期:第84-86页.

4 田云等.保护方式下的80386及其编程.北京:清华大学出版社,1993.12


# 博弈树搜索与静态估值函数


肖齐英 王正志

(长沙国防科大自动控制系 410073)



摘 要 本文就中国象棋残局这种博弈树来讨论博弈树的搜索技术和静态估值函数的关系。首先提出一个负极大值搜索算法,然后根据静态估值函数来讨论搜索算法求解效率。



关键词 负极大值 静态估值函数 理想估值函数


## 1 前言

目前,国际象棋程序达到甚至超过了世界冠军的水平,可见博弈树求解在这一领域的极大成功。然而,其它博弈如中国象棋、围棋却相差一段不小的距离。博弈一般分为开局、中局和残局。本文不论述开局的匹配搜索技术,也不研究中局短期,尤其是中局中长期策略,中局中长期策略是一个相当困难和十分复杂的问题。本文单就中国象棋残局这一特殊领域来研究博弈树的求解——博弈树搜索与静态估值函数的关系。

中国象棋残局(以下简称残局)的特点是搜索分枝比开局、中局都有所下降,大约20至30,博弈树虽然也是很庞大的。但理论上已经确定有唯一的正解,表现为红先胜、红先和,排局一般只有这两种结果,但实战残局还有红先负这种情况。残局阶段,先手、攻势、杀势尤为重要,往往一个先手可以克敌致胜,或者瓦解对方的杀势和攻势。复杂的残局再现了中局激烈拼杀的场面,而理论上的例胜,例和也要求有开局时的匹配搜索算法。因此对残局的研究体现了全局的思想。

## 2 搜索算法

搜索算法是博弈树求解的灵魂,只有有了有效的搜索算法才能在有限的时间内找到正确的解。

博弈树大都是相当庞大的,产生整个博弈树几乎


收稿日期:1997年4月26日

不可能,因此只可能搜索到一定的深度。若博弈树的分枝数为 B,产生深度为 D的博弈树,则有 $(B^{D+1}-1)/(B-1)$个结点,总结点数呈指数爆炸。搜索算法必须有效地减少产生的结点数才能提高搜索效率。现在我们来讨论负极大值算法。

文献[1][2]中均提出了负极大值算法。该算法的搜索结果等价于极小极大算法搜索所产生的结果。但显著的差别在于大大减少了搜索的结点数(在理想情况下)。但我们发现该算法在裁剪方面并不理想。在裁剪方面未能达到最佳裁剪。现在分析其原因: αβ裁剪技术理论告诉我们,MAX结点的 α值永不减少,而 MIN结点的 β值永不增加。根据这个理论,各层的 α值 β值都对直接下层裁剪起很重要作用。而该算法在裁剪时仅考虑了三级之间的裁剪。

因此,当超过三级时,尽管各结点的最左子女的裁剪频繁发生,而该算法却无能为力,多级之间应该裁剪的未能裁剪掉。

在最理想情况下,应该能裁剪掉 TOTAL- NED个结点,其中:

TOTAL= $(B^{D+1}-1)/(B-1)$

D为偶数时, NED= $2B^{D/2}-1$

D为奇数时, NED= $B^{(D+1)/2}+B^{(D-1)/2}-1$

但该算法未能裁剪掉这样多的结点数。考虑到多级之间的裁剪,我们重新设计了负极大值博弈树搜索算法。算法如下:

负极大值博弈树结点求值

Veb(x, player, l, alpha, beta, value, best)

1. [有必要构造博弈树吗?]

若 X是代表终局状态的结点或 l= 0

则 (1)若 player= 1

则 value<- E(x);

否则 value<- -E(x);

(2) best<-空棋盘状态;

(3) [算法结束]

2. [与已知棋局匹配, PB为已知棋局库]

若 X= Y(Y∈ PB)

则 value= Y的估计值;

[算法结束]

3. [产生下一步所有可能状态并排序用链表链起来]

调用 genernate(x, player, head);

4. [主程序调用时若只有唯一一步则不必估值返回该步]

若 head↑.next= NIL且 l= 10

则 best<- P;

[算法结束].

5. [准备执行负极大过程]

P<-head;

value<- beta;

best<- p;

6. [负极大过程]

循环当 p≠ NIL且 (value< alpha)

(1)调用 Veb(p,- player, l- 1,- value,- alpha, val, bes)

(2)若 (value<- val)

则 value<- -val;

best <- p

(3) p<- p↑.next

7. [释放空间]

(1) P<- head;

(2)循环当 P≠ NIL且 P≠ best

q<- p↑.next;

释放结点 P;

P<- q;

(3) P<- best↑.next;

best.next<- NIL;

(4)循环当 p≠ NIL

q<- p↑.next;

释放结点 p;

p<- q;

8. [算法结束]

在该算法中,MAX祖先结点对 MAX结点的裁剪通过将 MAX祖先值(alpha)下传到该 MAX结点的祖父 MAX结点而实现的; MIN祖先结点对 MIN结点的裁剪通过将 MIN祖先值(beta)下传到该 MIN结点的祖父 MIN结点而实现的。从而克服了文献[1][2]中的负极大值搜索算法只有祖父结点才对该结点有裁剪作用的不足。并保证裁剪是完备的。

用负极大值博弈树搜索算法时,虽然 α裁剪 β裁剪统一为一种裁剪,但为了方便起见,我们还按极小极大值算法时的 α裁剪 β裁剪分别区分它们。

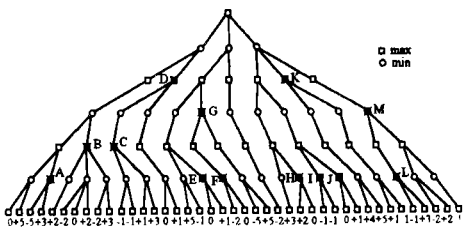


**图 1 多级裁剪**

例如对图 1所示的博弈树,我们考察了文献[1][2]中的负极大值搜索算法,它们仅对图中 A E G H进行了 α裁剪和对 B D与 K进行了 β裁剪,而对 G F I J L与 M没有进行恰当的裁剪,从而产生了过多的无用结点,影响了算法求解的效率。而用本文提出的算法不仅成功地对图中 A E G H进行了 α裁剪和对 B D与 K进行了 β裁剪,而且在 C处,按我们提出的

算法,alpha= 0,beta= -∞,value= 2,不满足(value < alpha)继续循环的条件,从而在 C处进行裁剪,同理还在 F J J L M 处进行了α裁剪。因此该算法比文献[1][2]中的相应算法的解题效率高。

## 3 静态估值函数

对于博弈树求解有了良好的搜索算法还只是问题的一个方面,问题的另一个方面就是静态估值函数。只有有了良好的静态估值函数才能保证较快地找到正解。而静态估值函数是对棋局的综合评估,该函数的好坏直接决定解题能力强与弱。通常一个优秀的棋手总有一个良好的对棋局的判断能力,能够协调各棋子的关系、取舍,有机地组织各棋子的进攻步调,控制棋局的发展。如果要把这一整套的思维物化成一个数值函数来评估,本身就是一个相当复杂的问题。有时一个损失极大的走步,在若干步之后体现出强大的优越性,有时一个看似取得明显实利的走步,却在对方步步紧逼之下变得极为不利。怎样评估一个棋局的优越性?

在本文中仅考虑残局(包括排局),首先各棋子本身的子力价值是不一样的。例如车的攻杀力最强,其子力价值应该最大,可设定为 100,而马攻杀力不及车与炮,但有时比炮强,可以设定为一个子力价值范围 45 - 50,炮攻杀力较强,但若无子配合时,攻击力大大减弱,也设定一个范围 50- 45,兵也有一定的攻击力,尤其是残局阶段多数强子已经牺牲,兵的威力渐渐显示出来,子力价值范围设为 10- 30等等。然后我们来对形势来评估,对形势的判断以及对进攻有效性的判断是关键。我们把攻击九宫,尤其是控将、叫将、叫杀、解杀点、扰乱对方进攻以及吃子、抽子与捉子等等形势判断出来并定以一定的局势分(1- 500)。然后把双方的局势分和子力价值分相加得出各自的棋局分。双方棋局分相减得出 e(x)。

当然对 e(x)的计算方法可以是多种多样的,但不管怎样设计 e(x),都难免有山脊、平原和小丘现象出现,这些现象的产生是问题本身所具有的,估值函数 e(x)无法克服,这就要通过搜索算法来解决。

## 4 搜索算法与估值函数结合

e(x)产生了本身无法克服的山脊、平原和小丘现象,而导致了对结点估值不真实,从而影响了结点的排序。而负极大值搜索算法受结点次序的影响很大。

既然裁剪与结点次序有极大的关系,我们就利用静态估值函数对产生的结点进行估值,并按有利于裁剪的次序进行排序。这样可以大大提高裁剪效率。在一定的时空条件下,可以适当加大搜索深度,从而又可以反过来解决部分的山脊、平原和小丘现象。当然要彻底地解决山脊、平原和小丘现象必须把搜索深度设成足够大,然而由于时间与空间的限制,这往往是不可能的。

对于博弈树搜索要每次达到理想的裁剪情况,就需要有准确无误的静态估值函数,然而设计这样的静态估值函数是不可能的,假如可能,就不要用搜索算法了,而直接用估值函数取其走步为最大值的那一步即可。

我们假设有一个估值函数使结点排序达到裁剪的理想情况,称这个估值为理想估值函数。我们只可能逼近这个理想估值函数,而不可能达到。对裁剪而言,我们设计的静态估值函数越接近这个理想估值函数,则裁剪掉的结点就越多,从而搜索深度就越大,山脊、平原和小丘现象就可多消除一些。然而这也是有限的,因为即使在最理想的裁剪情况下,在同一时间与同一空间的约束下,搜索深度也只能达到极小极大算法搜索深度的 2倍。当然对于多数残局问题已经可以解决了,但对于较为复杂的残局(类似于中局),这样的深度也难以彻底消除棋局固有的山脊、平原和小丘现象。这时负极大值搜索算法和静态估值函数仍无能为力,这是问题的固有难度所决定的,也证明了博弈仍然是一项复杂性极高的工作。

## 5 结论

本文提出了另一负极大值博弈树搜索算法,该算法克服了文献[1][2]中提出的负极大值博弈树搜索算法的不足,裁剪效率得到了极大的提高,搜索时间大大缩短。然后对静态估值函数进行了定性的分析,并与负极大值博弈树搜索算法结合,使裁剪结点尽量接近理想情况,从而可加大搜索深度,反过来部分地克服了静态估值函数的山脊、平原和小丘现象。在实践中用程序实现了上述思想,并对大量残局进行了解答,结果正确。

## 参考文献

1 陆汝钤《人工智能》
2 许卓群《数据结构》
3 Nils J. Nilsson Principles of Artificial Intelligence
4 Patrick Henry Winston Artificial Intelligence