**人工智能实验报告**

-----α-β剪枝实现一字棋博弈树搜索

小组成员：

06374048 张 曦

06374029 梁运德

06382091 张 帅

06382047 江金陵

α-β剪枝实现一字棋博弈树搜索

日期：2008年11月8日

1. **摘要：**

利用基于极大极小分析法的α-β剪枝搜索算法实现一字棋博弈树搜索算法优化。

1. **导言：**

问题描述：

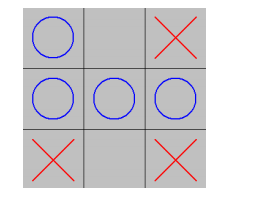
“一字棋”游戏（又叫“三子棋”或“井字棋”）， “井字棋”的棋盘很简单，是一个3×3的格子，“井字棋”是一方首先三子连成一线就胜利。使用的求解方法可以利用简单的宽度或深度优先搜索，而目前最为流行的就是这里采用的α-β剪枝搜索算法。

**三**．**实验过程：**

1. 算法设计：

**1.1** 极小极大分析法

设有九个空格，由MAX，MIN二人对弈，轮到谁走棋谁就往空格上放一只自己的棋子，谁先使自己的棋子构成“三子成一线”(同一行或列或对角线全是某人的棋子)，谁就取得了胜利。 用圆圈表示MAX，用叉号代表MIN。 比如下图中就是MAX取胜的棋局。



**1.2**估价函数定义如下：

设棋局为P，估价函数为e(P)。 (1) 若P对任何一方来说都不是获胜的位置，则e(P)=e(那些仍为MAX空着的完全的行、列或对角线的总数)-e(那些仍为MIN空着的完全的行、列或对角线的总数) (2) 若P是MAX必胜的棋局，则e(P)＝+∞ 。 (3) 若P是B必胜的棋局，则e(P)＝-∞。

**1.3 α-β**剪枝算法

上述的极小极大分析法，实际是先生成一棵博弈树，然后再计算其倒推值，至使极小极大分析法效率较低。于是在极小极大分析法的基础上提出了α-β剪枝技术。 α-β剪枝技术的基本思想或算法是，边生成博弈树边计算评估各节点的倒推值，并且根据评估出的倒推值范围，及时停止扩展那些已无必要再扩展的子节点，即相当于剪去了博弈树上的一些分枝，从而节约了机器开销，提高了搜索效率。具体的剪枝方法如下： (1) 对于一个与节点MIN，若能估计出其倒推值的上确界β，并且这个β值不大于 MIN的父节点(一定是或节点)的估计倒推值的下确界α，即α≥β，则就不必再扩展该 MIN节点的其余子节点了(因为这些节点的估值对MIN父节点的倒推值已无任何影响 了)。这一过程称为α剪枝。 (2) 对于一个或节点MAX，若能估计出其倒推值的下确界α，并且这个α值不小于 MAX的父节点(一定是与节点)的估计倒推值的上确界β，即α≥β，则就不必再扩展该MAX节点的其余子节点了(因为这些节点的估值对MAX父节点的倒推值已无任何影响 了)。这一过程称为β剪枝。 从算法中看到： (1) MAX节点(包括起始节点)的α值永不减少； (2) MIN节点(包括起始节点)的β值永不增加。 在搜索期间，α和β值的计算如下： (1) 一个MAX节点的α值等于其后继节点当前最大的最终倒推值。 (2) 一个MIN节点的β值等于其后继节点当前最小的最终倒推值。

**1.4** 输赢判断算法设计

因为每次导致输赢的只会是当前放置的棋子,输赢算法中只需从当前点开始扫描判断是否已经形成五子。对于这个子的八个方向判断是否已经形成五子。如果有，则说明有一方胜利，如果没有则继续搜索，直到有一方胜利或者搜索完整个棋盘。

1. 程序流程图：

比赛结束（胜负平）？

N

玩家下棋

N

Y

比赛结束（胜负平）？

调用alpha-belta函数进行搜索计算下一步落子位置

选择玩家先还是用户先

电脑先

显示初始棋盘

比赛结束提示结果信息

3.程序分析：

Ⅰ.实验环境：visual studio 2008

Ⅱ. 本次实验中我们采用了深搜两层的策略，因为深搜四层的话会造成评估函数大部分为0，而博奕中必须要搜索双数层因此取两层为佳。

Ⅲ.本次实验中设置了较好的人机交互以及出错检测。

4.程序主要函数：

**4．1**//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

//\* 电脑走一步棋。 \*

//\* 通过α-β剪枝搜索，找到电脑对电脑最有利的一步棋 \*

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

void search(node &father){

int em;

em=c\_empty(father);

if(em==0)return ;//当棋盘已满时，直接返回；

if(em==1){//棋盘只剩一个空格时，找到该空格，电脑填充空格,返回；

for(int i=0;i<3;i++){

for(int j=0;j<3;j++){

if(father.table [i][j]==empty){

father.table [i][j]=computer;

return;

}

}

}

}

else{//棋盘有两个以上，搜索对电脑最有利的一步棋；

vector<node> s;//存储电脑可以走的空格；

int alpha,beita;//极大、极小值；

node son;

for(int i=0;i<3;i++){//找出所有可以走的空格；

for(int j=0;j<3;j++){

if(father.table [i][j]==empty){

son=father;

son.table [i][j]=computer;

s.push\_back (son);

}

}

}

node n\_son;

alpha=-100;

bool expand;

int xia\_b;//指向下一步要走的空格；

for(int fp=0;fp<s.size ();fp++){//扩展每一个子节点；

beita=100;

expand=true;

for(int i=0;i<3;i++){

for(int j=0;j<3;j++){

if(s[fp].table [i][j]==empty){

n\_son=s[fp];

n\_son.table [i][j]=player;

n\_son.get\_value ();

/\*if(alpha<n\_son.value)

alpha=n\_son.value;

if(alpha>=beita)

expand=false;\*/

if(beita>n\_son.value )//获取一个子节点的极小值；

beita=n\_son.value ;

if(beita<=alpha){//如果极小值比父节点的极大值小，则不再扩展此节点；即进行α-β剪枝

expand=false;

break;

}

}

}

if(!expand)

break;

}

if(beita>alpha){//更新父节点的极大值；

alpha=beita;

xia\_b=fp;//更新指向下一步要走的空格的指针；

}

}

father=s[xia\_b];//电脑走一步棋；

}

return ;

}

**4．2**//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

//\*判断棋局的结果，如果电脑获胜返回，玩家获胜返回，和棋返回，还未能分出胜负返回-1；\*

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

int jude(int t[3][3]){

int a,b,c;

bool heqi=true;

for(int i=0;i<8;i++){

a=line[i][0];

b=line[i][1];

c=line[i][2];

if((t[a/3][a%3]==computer)&&(t[b/3][b%3]==computer)&&(t[c/3][c%3]==computer)){//一条直线上都是电脑的棋，电脑获胜；

return 2;//电脑胜

}

if((t[a/3][a%3]==player)&&(t[b/3][b%3]==player)&&(t[c/3][c%3]==player)){//一条直线上都是玩家的棋，玩家获胜；

return 1;//玩家胜

}

}

for(int i=0;i<3;i++)//判断棋盘是否已经下满个棋，如果没有下满，hepi的值赋为false；

for(int j=0;j<3;j++)

if(t[i][j]==empty)

heqi=false;

if(heqi){//如果棋盘下满，则棋局结果为和棋；

return 0;//和棋

}

return -1;//未分出胜负，继续下棋

}

**4．3** void get\_value(){ //获取评估值；

int a,b,c;

value=0;

for(int i=0;i<8;i++){

a=line[i][0];

b=line[i][1];

c=line[i][2];

if((table[a/3][a%3]==computer)&&(table[b/3][b%3]==computer)&&(table[c/3][c%3]==computer)){//电脑获胜，评估值为；

value=100;

return;

}

if((table[a/3][a%3]==player)&&(table[b/3][b%3]==player)&&(table[c/3][c%3]==player)){//电脑输，评估值为-50；

value=-50;

return;

}

if((table[a/3][a%3]!=player)&&(table[b/3][b%3]!=player)&&(table[c/3][c%3]!=player)){//一条直线上可能放的都是电脑的棋，评估值加；

value++;

}

if((table[a/3][a%3]!=computer)&&(table[b/3][b%3]!=computer)&&(table[c/3][c%3]!=computer)){//一条直线上可能放的都是玩家的棋，评估值减；

value--;

}

}

}

**四．实验小结：**

1. 通过本次实验进一步对老师课堂上所讲的AlphaBeta剪枝有了更加深刻的了解，对它的一般实现有了初步的认识。