**C语言项目报告**

**项目名称：五子棋**

**项目成员：佟新雨**

**填写日期：2020/9/9**

1. **摘要（Abstract）**

本项目主要内容在于设计一款能和五子棋禁手规则结合的五子棋AI，该AI具有一定的强度，可以胜过绝大多数普通五子棋玩家。与普通AI只能算出唯一解不同的是，这款AI具有一定的随机性，即对于价值相近的点都为最优解，并且随机落子。

1. **问题描述（Problem Statement）**

五子棋由于规则简单，容易上手，老少皆宜，被大家广泛的接受。但大多数人只是用于休闲娱乐，很少有人真正的把它当做像围棋，象棋的棋类运动。

这个项目的目的就是设计一个面向对五子棋有深入研究的五子棋爱好者的AI。这是因为常规的五子棋AI难度过低，而且不能与禁手规则结合。

五子棋不能登上正式场合的原因是其本身规则的缺陷，导致先行的一方（黑棋）必胜，而且是可以通过人脑计算出的。

这就导致了出现了很多规则，包括禁手规则（先行一方禁止使用的战术）。这里采用了最常见的一种禁手规则（双三，双四，长连禁手）。

1. **组内分工（Group Division）**

组长：佟新雨，承担全部工作

组员：（无）

1. **分析（Analysis）**

禁手规则主要是判断落下后的点的基本子力要素，由于基本子力要素是判断棋局的重要方式，故也将用于AI的设计中。

由于部分点基本子力要素与其实际价值不符，或基本子力要素中相同的要素价值不同，故自行设计了价值子力要素，将基本子力要素进行了更细的区分，并将原本的基本子力要素命名为规则子力要素。

AI能够计算出必胜下发是必须的，故ｎ步内必胜是AI算法的重要组成部分。如果能够计算出必胜，则AI落点会非常好。但是如果不能计算出ｎ步内必胜则需要比较每个点的价值，选出最优点。而最优点的难点在于如何比较每个点的价值。

1. **设计（Design）**

AI是自主设计，没有借鉴其他AI思路。计算机与人脑相比，最大的优势是其计算量大，计算速度快，要充分发挥这个优势。

故希望设计一个函数，计算出几步之内所有可能的情况，并寻找出必胜的点。因为棋盘上有２２５个点，如果计算５步所有可能情况大概需要计算１０＾１２～１４次，计算量过大，耗时过长，而且５步内人脑也很容易计算出来。故需要对其进行选择，不能所有的点都考虑进来。

结合人脑计算是否必胜的方式，发现人脑运算非常高效。初步设计了一个方案来对其进行优化。优化分两部分：

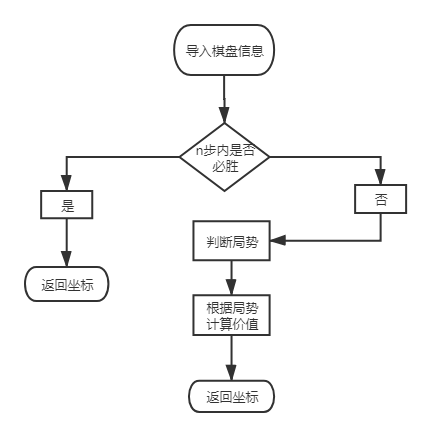
1. 判断什么情况可能会有必胜，因为有些情况显然不能必胜或几步之内不能必胜可能需要很多步。
2. 判断该点是否值得去下，如果不值得去下则不考虑落该点，例如棋盘的四角是不可能成为必胜方案中会用到的点的。

判断的步数显然要高于常人所能计算的步数，但又不能太高导致计算时间过长，最后经过测试发现计算１３步内必胜步数足够多而且计算最长时间控制在１０秒左右，是最优的选择。

AI统一用一个算法：寻找最优解算法，不同难度的算法主要区别在于计算必胜的步数不同，价值的计算方式不同。

寻找最优解算法首先判断是否必胜，如果不必胜则比较价值。这里采用 了本人平时下棋的具体思路和思考方式，而没有采用大量计算的方式。

具体为先判断棋局的局势，不同局势有不同的比较价值的方式。大致按 照棋局进度（开局，中期，后期），先后手，子力要素优先级，攻守形势等 划分不同的比较价值的方式。这里会有一个简单的判断，如果对方３步内必 胜则此点的价值会发生很大的变化。这样做弥补了部分计算ｎ步必胜算法时 因为优化而导致的强度的降低。



1. **实施（Implementation）**

基本子力要素的具体实现代码：

以四的规则子力要素为例。

int length( int a,int b,board \*pboard,int mode, //求(a,b)点连珠长度

int \*frontx=NULL,int \*fronty=NULL, //并返回两端(包含棋子)坐标

int \*rearx=NULL,int \*reary=NULL){

//工作变量

int colour=pboard->everypoint[a][b].color; //所求的棋子颜色

int px,py,qx,qy; //指针

bool f; //判断是否停止

int l; //记录连珠长度

int x,y; //记录工作方向

switch(mode){

case 0:x=0;y=1;break; //从上到下

case 1:x=1;y=1;break; //从左上到右下

case 2:x=1;y=0;break; //从左到右

case 3:x=1;y=-1;break; //从左下到右上

}

px=a;py=b;qx=px-x;qy=py-y;f=true;l=0;//初始化工作变量

//移动指针到最上端

while(f){

if(qx>14||qx<0||qy>14||qy<0)

f=false; //如果超出边界结束

else

if(pboard->everypoint[qx][qy].color!=colour)

f=false;//如果没有子或颜色不同结束

else{

qx=qx-x;qy=qy-y;px=px-x;py=py-y;//移动指针

}

}

qx=px;qy=py; //结束时px，py应指向一段

//检测长度l

while(qx>=0&&qx<15&&qy>=0&&qy<15&&pboard->everypoint[qx][qy].color==colour) {//当颜色相同且没有出界时

qx=qx+x;qy=qy+y; //将qx，qy移向另一端 (最后在连珠外一格)

l++;

}

qx=qx-x;qy=qy-y; //归位

//返回两端坐标

if(frontx!=NULL&&fronty!=NULL){

\*frontx=px;\*fronty=py;

}

if(rearx!=NULL&&reary!=NULL){

\*rearx=qx;\*reary=qy;

}

return l; //返回连珠长度l

}

int five(int a,int b,board \*pboard){ //求（a，b）四个方向形成"5"的个数

int i=0,l=0;

int n=0; //记录4个方向 形成"5"的个数

for(i=0;i<4;i++){

l=length(a,b,pboard,i);

if(pboard->gamemode.forbidden\_moves==1&&pboard->everypoint[a][b].color==1){//有禁手

if(l==5) n++;

}else{ //无禁手

if(l>=5) n++;

}

}

return n;

}

int is\_four(int a,int b,board \*pboard,int mode,

/\*判断mode方向是否是4，活四返回2及两端坐标，冲四返回能成5端坐标,没有4则返回0,若能形成两个冲四则返回3\*/

int \*px1=NULL,int \*py1=NULL,int \*px2=NULL,int \*py2=NULL){//若不能成5或在棋盘外坐则标返回为-1

int px,py,qx,qy;//工作指针，标志两端坐标

int n=0; //判断"4"类型

length(a,b,pboard,mode,&px,&py,&qx,&qy); //求两端坐标

int x,y; //记录工作方向

switch(mode){

case 0:x=0;y=1;break; //从上到下

case 1:x=1;y=1;break; //从左上到右下

case 2:x=1;y=0;break; //从左到右

case 3:x=1;y=-1;break; //从左下到右上

}

px=px-x;py=py-y;qx=qx+x;qy=qy+y;//初始化工作变量 (即将指针向外移一格)

board board\_1,board\_2;

board\_1=\*pboard;board\_2=\*pboard; //两种情况的棋盘信息

//第一种情况 (board\_1)

if(px>=0&&px<15&&py>=0&&py<15&&pboard->everypoint[px][py].color==-1){ //如果端点在棋盘内并且为没有子

board\_1.everypoint[px][py].color=pboard->everypoint[a][b].color; //将board1情况的一端落相同颜色的子

if(pboard->gamemode.forbidden\_moves==1&&pboard->everypoint[a][b].color==1){//有禁手

if(length(a,b,&board\_1,mode)==5){ //判断是否形成5,若形成n+1

n++;

if(px1!=NULL&&py1!=NULL){ //若形成5返回一端坐标

\*px1=px;\*py1=py;

}

}else{

if(px1!=NULL&&py1!=NULL){ //若没有形成5返回-1

\*px1=-1;\*py1=-1;

}

}

}else{ //无禁手

if(length(a,b,&board\_1,mode)>=5){ //判断是否形成5,若形成n+1

n++;

if(px1!=NULL&&py1!=NULL){ //若形成5返回一端坐标

\*px1=px;\*py1=py;

}

}else{

if(px1!=NULL&&py1!=NULL){ //若没有形成5返回-1

\*px1=-1;\*py1=-1;

}

}

}

}

else{

if(px1!=NULL&&py1!=NULL){ //若在界外或该点有子返回-1

\*px1=-1;\*py1=-1;

}

}

//第二种情况(board\_2)

if(qx>=0&&qx<15&&qy>=0&&qy<15&&pboard->everypoint[qx][qy].color==-1){ //如果端点在棋盘内并且为没有子

board\_2.everypoint[qx][qy].color=pboard->everypoint[a][b].color; //将board2情况的一端落相同颜色的子

if(pboard->gamemode.forbidden\_moves==1&&pboard->everypoint[a][b].color==1){//有禁手

if(length(a,b,&board\_2,mode)==5){ //判断是否形成5,若形成n+1

n++;

if(px2!=NULL&&py2!=NULL){ //若形成5返回一端坐标

\*px2=qx;\*py2=qy;

}

}else{

if(px2!=NULL&&py2!=NULL){ //若没有形成5返回-1

\*px2=-1;\*py2=-1;

}

}

}else{ //无禁手

if(length(a,b,&board\_2,mode)>=5){ //判断是否形成5,若形成n+1

n++;

if(px2!=NULL&&py2!=NULL){ //若形成5返回一端坐标

\*px2=qx;\*py2=qy;

}

}else{

if(px2!=NULL&&py2!=NULL){ //若没有形成5返回-1

\*px2=-1;\*py2=-1;

}

}

}

}else{

if(px2!=NULL&&py2!=NULL){ //若在界外或该点有子返回-1

\*px2=-1;\*py2=-1;

}

}

if(n==2) //特殊情况

if(length(a,b,pboard,mode)<4)

n++;

return n;

}

void four( int a,int b,board \*pboard, //求（a，b）四个方向活四和冲四的个数

int \*plivefour,int \*potherfour){

int i=0;//标志方向

int kind\_four=0;//标志4的种类

for(i=0;i<4;i++){

kind\_four=is\_four(a,b,pboard,i);

if(kind\_four==2)

\*plivefour=\*plivefour+1;

if(kind\_four==1)

\*potherfour=\*potherfour+1;

if(kind\_four==3)

\*potherfour=\*potherfour+2;

}

}

ｎ步必胜算法：

int steps\_to\_win(board \*pboard,int n,int \*a=NULL,int \*b=NULL){

if(n==1){

int m;

m=sure\_to\_win(pboard,a,b);

return m;

}else{

board board\_1,board\_2;

int f,g,h,x,y;

bool forbidden\_moves,forbidden\_moves\_2;

bool can\_go=false;//判断己方落完子后对方能否落子

int i,j,k,l;

int pa=-1,pb=-1;

f=sure\_to\_win(pboard,a,b);

if(f==1){ //若单步必胜返回坐标

//printf("!\n!\n!\n%d:(%d,%d)1步必胜\n",n,\*a,\*b);

return 1;

} //并结束计算

for(i=0;i<15;i++) //判断对方是否有四（即对方能否成五） ，若能成五结束计算

for(j=0;j<15;j++){

f=can\_he\_form\_five(i,j,pboard);

if(f!=0){

pa=i;pb=j;

if(a!=NULL&&b!=NULL){

\*a=pa;\*b=pb;

}

//printf("%d:对方有四\n",n);

return 0;

}

}

for(i=0;i<15;i++) //判断己方是否有活三（即能成活四），若有则落该点并返回该点坐标

for(j=0;j<15;j++){

f=can\_form\_livefour(i,j,pboard);

if(f!=0){

if(pboard->gamemode.forbidden\_moves==1&&(pboard->number%2)==0){//如果是黑棋，有禁手

g=can\_form\_otherfour(i,j,pboard);

h=can\_form\_livethree(i,j,pboard);

x=can\_form\_overfive(i,j,pboard);

if((f+g)<2&&h<2&&x==0){//如果落下后形成四但不是禁手,则返回坐标

pa=i;pb=j;

if(a!=NULL&&b!=NULL){

\*a=pa;\*b=pb;

}

//printf("!\n!\n!\n%d:(%d,%d)两步必胜\n",n,pa,pb);

return 1;

}

}else{

pa=i;pb=j;

if(a!=NULL&&b!=NULL){

\*a=pa;\*b=pb;

}

//printf("!\n!\n!\n%d:(%d,%d)两步必胜\n",n,pa,pb);

return 1;

}

}

}

for(i=0;i<15;i++) //判断对方能否形成活四

for(j=0;j<15;j++){

f=can\_he\_form\_livefour(i,j,pboard);

if(f!=0){

if(pboard->gamemode.forbidden\_moves==1&&(pboard->number%2)==1){//如果是黑棋，有禁手

g=can\_he\_form\_otherfour(i,j,pboard);

h=can\_he\_form\_livethree(i,j,pboard);

x=can\_he\_form\_overfive(i,j,pboard);

if((f+g)<2&&h<2&&x==0){//如果落下后形成四但不是禁手,则结束计算

if(a!=NULL&&b!=NULL){

\*a=pa;\*b=pb;

}

//printf("%d:对方有活三（己方没有活三,双方都没有四）\n",n);

return 0;

}

}else{

if(a!=NULL&&b!=NULL){

\*a=pa;\*b=pb;

}

//printf("%d:对方有活三（双方都没有四）\n",n);

return 0;

}

}

}

for(i=0;i<15;i++) //遍历所有点，判断能否形成活三或冲四

for(j=0;j<15;j++){

forbidden\_moves=can\_form\_forbidden\_moves(i,j,pboard);

if(!forbidden\_moves){//如果落下后没有形成禁手

f=can\_form\_livethree(i,j,pboard);

g=can\_form\_otherfour(i,j,pboard);

if(f>0||g>0){//如果可以形成活三或冲四

//落己方子

board\_1=\*pboard;

board\_1.number++;

board\_1.everypoint[i][j].color=board\_1.number%2;

board\_1.everypoint[i][j].number=board\_1.number;

//落对方子

if(g>0){//如果己方落的是冲四点

for(k=0;k<15;k++) //判断每个点己方是否成五

for(l=0;l<15;l++){

h=can\_he\_form\_five(k,l,&board\_1);

if(h>0){//如果己方能成五

//如果对方是禁手

forbidden\_moves\_2=can\_form\_forbidden\_moves(k,l,&board\_1);

if(forbidden\_moves\_2){

pa=i;pb=j;

if(a!=NULL&&b!=NULL){

\*a=pa;\*b=pb;

}

//printf("!\n!\n!\n%d:(%d,%d)冲四逼禁手\n",n,pa,pb);

return 1;

}

//落对方子

board\_2=board\_1;

board\_2.number++;

board\_2.everypoint[k][l].color=board\_2.number%2;

board\_2.everypoint[k][l].number=board\_2.number;

//判断n-1步是否必胜

x=steps\_to\_win(&board\_2,n-1);

if(x==1){//如果n-1步必胜，返回（i，j）

pa=i;pb=j;

if(a!=NULL&&b!=NULL){

\*a=pa;\*b=pb;

}

//printf("!\n!\n!\n%d:(%d,%d)冲四获胜(如果n=2则是双四获胜)\n",n,pa,pb);

return 1;

}

}

}

//printf("%d:有己方冲四点(%d，%d)可落，但不能必胜\n",n,i,j);

}else{//如果落下后没有形成冲四但形成了活三

can\_go=false;

for(k=0;k<15;k++)//判断每个点己方能否成活四或对方能否成冲四

for(l=0;l<15;l++){

h=can\_he\_form\_livefour(k,l,&board\_1);

x=can\_form\_otherfour(k,l,&board\_1);

if(h>0||x>0){//如果己方能成活四或对方能成冲四

forbidden\_moves\_2=can\_form\_forbidden\_moves(k,l,&board\_1);

if(!forbidden\_moves\_2){//如果对方不是禁手

can\_go=true;//对方有可落点

//落对方子

board\_2=board\_1;

board\_2.number++;

board\_2.everypoint[k][l].color=board\_2.number%2;

board\_2.everypoint[k][l].number=board\_2.number;

//判断n-步是否必胜

y=steps\_to\_win(&board\_2,n-1);

if(y==1){//如果n-1步必胜

pa=i;pb=j;

if(a!=NULL&&b!=NULL){

\*a=pa;\*b=pb;

}

//printf("!\n!\n!\n%d:(%d,%d)可下但不可防的活三\n",n,i,j);

return 1;

}

}

}

}

if(!can\_go){

pa=i;pb=i;

if(a!=NULL&&b!=NULL){

\*a=pa;\*b=pb;

}

//printf("!\n!\n!\n%d:(%d,%d)不可下不可防的活三\n",n,i,j);

return 1;

}

}

}

}

}

if(a!=NULL&&b!=NULL){

\*a=-1;\*b=-1;

}

//printf("%d:未找到必胜点\n",n);

return 0;

}

}

寻找最优点算法：（函数过长，只截取一种局势的情况）

void best\_point(board \*pboard,int \*a,int \*b){

······

if(my\_points!=0||his\_points!=0){//如果存在可下点

if(pboard->number%2==0){//黑棋主进攻

if(my\_points!=0){

if((my\_points-his\_points)>=-2){//比较己方点价值

value=0;value\_max=0;

r=0;

for(i=0;i<15;i++)//遍历所有点

for(j=0;j<15;j++){

if(value\_of\_everypoint[i][j].number\_attack==1){//如果进攻点被激活

value=value\_of\_everypoint[i][j].value;

if(value>value\_max)//比较价值

value\_max=value;

}

}

for(i=0;i<15;i++)//给最大价值的点赋上序号

for(j=0;j<15;j++){

if(value\_of\_everypoint[i][j].number\_attack==1&&value\_of\_everypoint[i][j].value==value\_max){

//如果是进攻点并且价值最大

r++;

value\_of\_everypoint[i][j].random\_number=r;

}

}

random=rand();

r=random%r+1;//给r赋一个随机数

for(i=0;i<15;i++)//根据随机数返回坐标

for(j=0;j<15;j++){

if(value\_of\_everypoint[i][j].random\_number==r){

\*a=i;\*b=j;

}

}

return;

}else{//比较对方点价值

value=0;value\_max=0;

r=0;

for(i=0;i<15;i++)//遍历所有点

for(j=0;j<15;j++){

if(value\_of\_everypoint[i][j].number\_defence==1){//如果防守点被激活

value=value\_of\_everypoint[i][j].value;

if(value>value\_max)//比较价值

value\_max=value;

}

}

for(i=0;i<15;i++)//给最大价值的点赋上序号

for(j=0;j<15;j++){

if(value\_of\_everypoint[i][j].number\_defence==1&&value\_of\_everypoint[i][j].value==value\_max){

//如果是防守点并且价值最大

r++;

value\_of\_everypoint[i][j].random\_number=r;

}

}

random=rand();

r=random%r+1;//给r赋一个随机数

for(i=0;i<15;i++)//根据随机数返回坐标

for(j=0;j<15;j++){

if(value\_of\_everypoint[i][j].random\_number==r){

\*a=i;\*b=j;

}

}

return;

}

}else{//比较对方点价值

value=0;value\_max=0;

r=0;

for(i=0;i<15;i++)//遍历所有点

for(j=0;j<15;j++){

if(value\_of\_everypoint[i][j].number\_defence==1){//如果防守点被激活

value=value\_of\_everypoint[i][j].value;

if(value>value\_max)//比较价值

value\_max=value;

}

}

for(i=0;i<15;i++)//给最大价值的点赋上序号

for(j=0;j<15;j++){

if(value\_of\_everypoint[i][j].number\_defence==1&&value\_of\_everypoint[i][j].value==value\_max){

//如果是防守点并且价值最大

r++;

value\_of\_everypoint[i][j].random\_number=r;

}

}

random=rand();

r=random%r+1;//给r赋一个随机数

for(i=0;i<15;i++)//根据随机数返回坐标

for(j=0;j<15;j++){

if(value\_of\_everypoint[i][j].random\_number==r){

\*a=i;\*b=j;

}

}

return;

}

}else{//白

if(his\_points!=0){

if((his\_points-my\_points)>=-2){//比较对方点价值

value=0;value\_max=0;

r=0;

for(i=0;i<15;i++)//遍历所有点

for(j=0;j<15;j++){

if(value\_of\_everypoint[i][j].number\_defence==1){//如果防守点被激活

value=value\_of\_everypoint[i][j].value;

if(value>value\_max)//比较价值

value\_max=value;

}

}

for(i=0;i<15;i++)//给最大价值的点赋上序号

for(j=0;j<15;j++){

if(value\_of\_everypoint[i][j].number\_defence==1&&value\_of\_everypoint[i][j].value==value\_max){

//如果是防守点并且价值最大

r++;

value\_of\_everypoint[i][j].random\_number=r;

}

}

random=rand();

r=random%r+1;//给r赋一个随机数

for(i=0;i<15;i++)//根据随机数返回坐标

for(j=0;j<15;j++){

if(value\_of\_everypoint[i][j].random\_number==r){

\*a=i;\*b=j;

}

}

return;

}else{//比较己方点价值

value=0;value\_max=0;

r=0;

for(i=0;i<15;i++)//遍历所有点

for(j=0;j<15;j++){

if(value\_of\_everypoint[i][j].number\_attack==1){//如果进攻点被激活

value=value\_of\_everypoint[i][j].value;

if(value>value\_max)//比较价值

value\_max=value;

}

}

for(i=0;i<15;i++)//给最大价值的点赋上序号

for(j=0;j<15;j++){

if(value\_of\_everypoint[i][j].number\_attack==1&&value\_of\_everypoint[i][j].value==value\_max){

//如果是进攻点并且价值最大

r++;

value\_of\_everypoint[i][j].random\_number=r;

}

}

random=rand();

r=random%r+1;//给r赋一个随机数

for(i=0;i<15;i++)//根据随机数返回坐标

for(j=0;j<15;j++){

if(value\_of\_everypoint[i][j].random\_number==r){

\*a=i;\*b=j;

}

}

return;

}

}else{//比较己方点价值

value=0;value\_max=0;

r=0;

for(i=0;i<15;i++)//遍历所有点

for(j=0;j<15;j++){

if(value\_of\_everypoint[i][j].number\_attack==1){//如果进攻点被激活

value=value\_of\_everypoint[i][j].value;

if(value>value\_max)//比较价值

value\_max=value;

}

}

for(i=0;i<15;i++)//给最大价值的点赋上序号

for(j=0;j<15;j++){

if(value\_of\_everypoint[i][j].number\_attack==1&&value\_of\_everypoint[i][j].value==value\_max){

//如果是进攻点并且价值最大

r++;

value\_of\_everypoint[i][j].random\_number=r;

}

}

random=rand();

r=random%r+1;//给r赋一个随机数

for(i=0;i<15;i++)//根据随机数返回坐标

for(j=0;j<15;j++){

if(value\_of\_everypoint[i][j].random\_number==r){

\*a=i;\*b=j;

}

}

return;

}

}

}

······

}

1. **测试（Test）**

功能测试：AI功能基本实现，具有一定的强度。

界面展示：（图中为两个AI进行的对局的结果，即人机对战+提示）