人工知能最終レポート

035743A：比嘉雅樹

1　課題

R.Axelrod の IPD コンテストに出場するための IPD ルールを設計し、評価せよ。利得テーブルは下記を使用するものとし、基本ルールは第一回大会に準ずるものとする。
評価については、TFT やランダム、他のレポート作成者の作成したルールとの対戦結果などを用いるとよい。

2　プログラム

課題で用いたプログラムを以下に示す。

```
import java.util.*;


class test{
public static void main(String args[]){
//ai_unit unit2 = new all_C();
//ai_unit unit2 = new all_D();
ai_unit unit1 = new original();
ai_unit unit2 = new TFT();
//ai_unit unit2 = new unit_rand();
//ai_unit unit2 = new mori();
//ai_unit unit2 = new yuji();
Axelrod a = new Axelrod(unit1,unit2);
a.run(200);
}
}

interface ai_unit{
void learnning(int i);//相手が入力してきたものを受け取る
int call();//次に出す者を戻り値として返す
}

//C=0,D=1
class original implements ai_unit{
```

```
int aite,counter=0,c=0;
public void learnning(int i){
aite=i;
}
public int call(){
counter++;
if(counter==1)
return 1;
if(aite==0) {
c++;
return 0;
}
else if(c%3 == 0){
return 1;
}
else {
return 1;
}

}
}

class mori implements ai_unit{

int aite,ucount=1;
public void learnning(int i){
aite=i;
}
public int call(){
if(aite==0)
return 0;
else
if(ucount%3==0)
return 1;
else {
ucount++;
return 0;
}
}
```

```
}


class yuji implements ai_unit{
int aite,ucount=1;
public void learnning(int i){
aite=i;
}
public int call(){
if(aite==0)
return 1;
else
if(ucount%2==0)
return 1;
else {
ucount++;
return 0;
}
}
}


class all_C implements ai_unit{//0=C,1=D
public void learnning(int i){
}
public int call(){
return 0;
}
}

class all_D implements ai_unit{//0=C,1=D
public void learnning(int i){
}
public int call(){
return 1;
}
}

class TFT implements ai_unit{
```

```
int counter=0; //call が呼ばれた回数
int memo; //前回相手の出したもの
public void learnning(int i){
memo=i;
}
public int call(){
counter++;
if(counter==1)
return 0;
else
return memo;
}
}

class unit_rand implements ai_unit{
int i;
Random r = new Random(1234);
public void learnning(int n){
i=n;
}
public int call(){
i=r.nextInt()%2;
return i*i;
}
}


class Axelrod{
ai_unit A,B;

int array1[][] ={{3,0},{5,1}};
int array2[][] ={{3,5},{0,1}};
int yellow=5,y_c=0;

Axelrod(ai_unit a,ai_unit b){
A=a;
B=b;
}
```

```
void run(int count){//i 回実行する
int u_a,u_b,i;
int asam=0,bsam=0;
int a_nice=count,b_nice=count;
for(i=0;i<count;i++){
u_a=A.call();
u_b=B.call();
a_nice-=u_a;
b_nice-=u_b;
/*計算*/
System.out.print("("+array1[u_a][u_b]+" ,  "+array2[u_a][u_b]+")\t");
asam+=array1[u_a][u_b];
bsam+=array2[u_a][u_b];
if(++y_c%yellow==0)
System.out.println();
A.learnning(u_b);
B.learnning(u_a);


}
System.out.println();
System.out.println("SAM->("+asam+","+bsam+")");
System.out.println("NICE->("+a_nice+","+b_nice+")");
}
}
```

プログラムの最初の方にある、unit1 と unit2 で勝負するプログラム。
unit1 が自分で考えたルールで、unit2 が対戦するルールである。実行すると勝負の様子が出力され、最終的に点数の合計が表示される。（左が自分、右が相手）NICE は協調した回数である。

3　対戦評価

TFT との評価

```
SAM->(500,500)
NICE->(100,100)
```

ランダムとの評価

```
SAM->(447,447)
NICE->(97,97)
```

他の作成者との比較

- もりとし

```
SAM->(602,597)
NICE->(199,200)
```

- ゆうじ

```
SAM->(203,203)
NICE->(1,1)
```

上の対戦結果を見てみると、勝利か引き分けしかなく、どうにかバランスのとれたルールが作れたと思う。