Práctica 3: Desconecta 4 Boom

Álvaro Beltrán Camacho

Índice

1. Introducción

En esta práctica tenemos que implementar una inteligencia artificial para el juego desconecta 4 Boom, donde tenemos un tablero 7x7, en el que no podemos hacer 4 en línea.Es un juego de dos jugadores (fichas verdes o azules) y podemos usar unas fichas bomba para eliminar nuestras fichas de esa fila.

2. Algoritmo

Para jugar a este juego hemos usado el algoritmo de poda alfa-beta. Que consiste en generar un árbol de posibilidades minimax y podar con respecto a los máximos y mínimos. Aporto pseudocódigo:

```
double alfabeta(node,jugador,depth,&accion,alfa,beta):

 if profundidad is 8 or Enviroment is terminal :
       return Valoracion(node,jugador)

 if porfundidad is pair :
    bestVal = -INFINITY

    for each node sucessor :
       value = alfabeta(sucessor,jugador,depth+1,&accion,alfa,beta)

       if value > bestVal:
          bestVal=value;
          if depth is 0 :
             accion = sucessor.Last_Action(jugador);

       alpha = max( alpha, bestVal)
       if beta <= alpha:
          break
    return bestVal

 else :
    bestVal = +INFINITY

    for each child node :
        value = alfabeta(sucessor,jugador,depth+1,&accion,alfa,beta)
        bestVal = min( bestVal, value)
        beta = min( beta, bestVal)
        if beta <= alpha:
                break
    return bestVal
```

Aparte de esta función el código aporta un metodo **think()** donde compruebo si

gano en alguna de las siguientes jugadas. Si gano; tomo esta accion, y si no, llamo a **alfabeta(node,jugador,depth,& accion,alfa,beta)**.

3. Heurística

Por último,voy a describir mi función heurística. En el juego nos encontramos tres casos diferenciales: si el jugador ha ganado, si ha perdido o si ha empatado. Mi decisión con respecto a esto fue que si ha ganado devolvemos el mayor valor posible, si ha perdido el peor valor posible y si ha empatado devolvemos 0. Pero esta no es una buena heurística porque no discrimina las opciones del empate.

Por tanto, nos queda estudiar que pasa si empata. Teniendo en cuenta que perder es hacer 4 en línea, voy a tomar como jugadas malas hacer 2 o 3 en línea. Para ello he creado las funciones **ValoracionHorizontalVertical(estado,jugador)** y **ValoracionDiagonal(estado,jugador)**.

La primera función estudia las líneas verticales y horizontales del tablero.Por cada 3 en línea suma 3000 si son del contrario o resta 3000 si son de jugador. Además, Por cada 2 en línea suma 150 si son del contrario o resta 150 si son de jugador.

La segunda función estudia las líneas diagonales del tablero.Por cada 3 en diagonal suma 3000 si son del contrario o resta 3000 si son de jugador. Además, Por cada 2 en diagonal suma 150 si son del contrario o resta 150 si son de jugador.

Una vez implementada esta heurística, ví que no discriminaba del todo los empates, pues había muchos con la misma valoración. Para solucionar esto, decidí sumarle al valor un número aleatorio entre 0 y 100 para que no siempre cogiese la primera acción encontrada con esa valoración. Escogí el intervalo [0,100] porque produce pequeños cambios ya que las otras funciones suman y restan como poco 150.

Presento el psudocódigo:

```
double Valoracion(estado,jugador) :

    double value=0;

    if jugador has won :
        value= +INFINITY
    else if jugador has tied :
        value+=ValoracionHorizontalVertical(estado,jugador);
        value+=ValoracionDiagonal(estado,jugador);
```

```
        value+=rand() % 100;
    else :
        value= -INFINITY;

    return value;
```