Memoria Práctica 3

Búsqueda con Adversario (juegos)

Inteligencia Artificial

Mario Ruiz Calvo (2º D)

Análisis del problema.

Se pide diseñar un agente deliberativo que permita llevar a cabo un comportamiento inteligente dentro del juego Conecta-4.

Descripción de la solución planteada.

Algoritmo MINIMAX con poda Alfa-Beta

Variables

Las variables que se han empleado para el desarrollo del algoritmo minimax con poda alfa-beta son:

- tablero: Variable de tipo *Environment* que representa el tablero 7x7 del conecta-4.
- jug: Variable enteras que representa el jugador activo que hace la llamada al algoritmo.
- prof: Variable entera que representa la profundidad en la que se está evaluando el algoritmo en cada momento.
- limi: Variable entera que representa la profundidad máxima hasta que la que va a explorar el algoritmo.
- accion: Variable de tipo *AccionType* que devuelve la acción que se ha decidido tomar tras la ejecución del algoritmo.
- a: Variable de tipo entero que representa la cota alfa utilizada para podar nodos MIN. (Se pasa por valor)
- b: Variable de tipo entero que representa la cota beta utilizada para podar nodos MAX. (Se pasa por valor)

El algoritmo devuelve el valor minimax del árbol de búsqueda.

Descripción del algoritmo

Al comenzar el algoritmo (profundidad = 0) se inicializan las cotas alfa y beta a menos infinito (-9999999999) y mas infinito (9999999999).

Si la profundidad no ha llegado al límite se van generando los hijos con profundidad +1. Si el nodo que se está evaluando es un nodo MAX (la profundidad es par). Si el valor minimax que devuelve el hijo es mayor que la cota alfa, se guarda el nuevo valor y se devuelve la acción. Además si la cota alfa es mayor a la cota beta se devuelve la cota beta sin tener que explorar sus hijos. Análogamente se procede si es un nodo MIN, cambiando alfa por beta.

Si la profundidad ha llegado al límite se evalúa la función heurística y se

devuelve su valor.

Función Valoración1. Heurísticas.

Esta función será la encargada de devolver un valor atendiendo a lo bueno que es un tablero para el jugador que la llama.

Para ello, en primer lugar, revisa que no haya acabado la partida. Si ha acabado devolverá 9999999999, -9999999999 o 0 dependiendo de si el jugador que ha llamado a la función a ganado, perdido o empatado.

Si la partida no ha terminado evaluará el tablero para el jugador que llama a la función y para el contrincante y resta los valores.

MiPuntuacion

Esta función será la encargada de evaluar el tablero para un determinado jugador. Para ello recorre el tablero y para cada posición si el valor coincide con el jugador que hace la llamada devuelve el valor de la heurística para esa posición que se irá acumulando hasta evaluar el tablero completo.

Heurísticas

Durante el desarrollo de la práctica se han tenido en cuenta diferentes heurísticas. Algunas de ellas son:

MiPuntuacionIJ

Evalúa un tablero 3x3 generado a partir del tablero 7x7 original y una posición dada de ese tablero. Para cada posición de ese tablero 3x3 se sumará un valor 2 si la casilla está ocupada por una ficha del jugador que ha hecho la llamada a la función y un 1 si la casilla esta libre.

MiPuntuacionIJ2

Evalúa un tablero 3x3 generado a partir del tablero 7x7 original y una posición dada de ese tablero. Devolverá un valor que representa el número de filas, columnas o diagonales en las que el jugador todavía puede poner tres fichas en línea, es decir, las filas, columnas o diagonales en las que no hay colocada ninguna ficha del jugador contrario. Como máximo puede devolver un 8, por las 3 filas, 3 columnas y 2 diagonales que puede tener libres.

MiPuntuacionIJ3

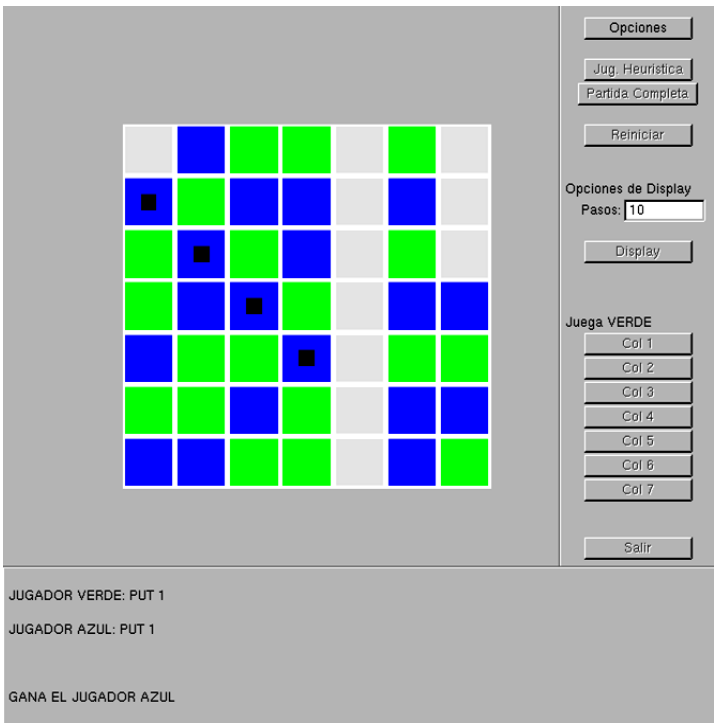
Igual que el anterior pero tomando como partida un tablero 5x5 en lugar de 3x3. Como máximo puede devolver un 12, por las 5 filas, 5 columnas y 2 diagonales que puede tener libres.

EnLinea3

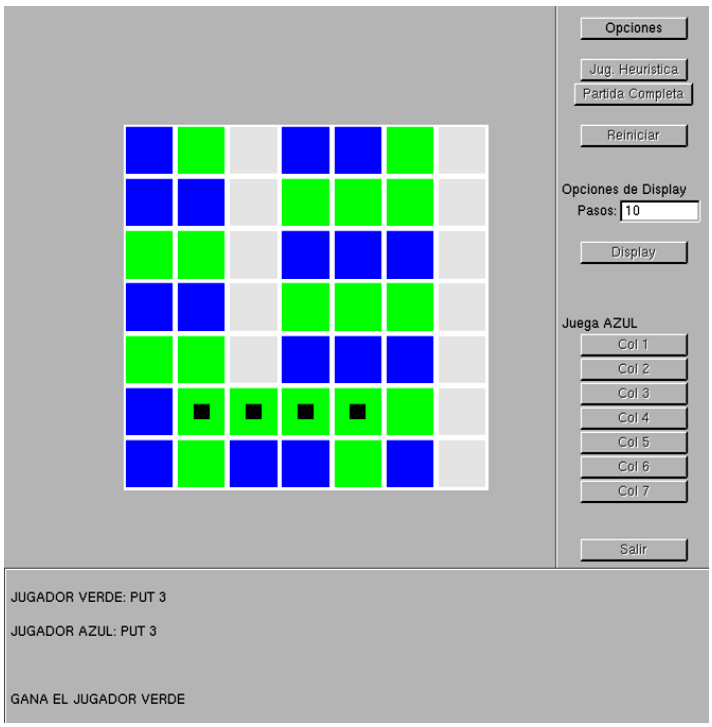
Determina si hay tres fichas del jugador en línea (ya sea en una fila, columna o en una diagonal) en una posición dada y devuelve un 2 si uno de los extremos esta libre, un 4 si los dos están libres y un 0 si no tiene ningún extremo libre con lo que no se puede buscar el cuatro en línea. También localiza posibles 4 en línea en una fila, columna o diagonal cuando la única casilla que queda para conseguir el 4 en línea es una de las dos casillas del centro, en este caso devuelve un dos.
Si se da más de uno de los casos al mismo tiempo, los valores se van acumulando.

Heurística elegida.

Tras una serie de pruebas realizadas para las heurísticas mencionadas anteriormente (incluyendo varias combinaciones de ellas), se ha decidido utilizar como heurística para la función MiPuntuacion la suma de las heurísticas MiPuntuacionIJ3 y EnLinea3



Heurística elegida (Azul) VS Ninja (Verde)



Heurística elegida (Verde) VS Ninja (Azul)

```
double MiPuntuacion(int jugador, const Environment &estado){

    double suma=0;

    for (int i=0; i<7; i++)
      for (int j=0; j<7; j++)
        if (estado.See_Casilla(i,j)==jugador)
              suma+=MiPuntuacionIJ3(i,j,jugador,estado)+EnLinea3(i,j,jugador,estado);

    return suma;
}
```