MARCOS AVILÉS LUQUE
Grupo: 8
Práctica 3

INTELIGENCIA ARTIFICIAL

E.T.S. de Ingenierías Informática y de Telecomunicación

Práctica 3

Métodos de Búsqueda con Adversario (Juegos)
DESCONECTA-4 BOOM

DEPARTAMENTO DE CIENCIAS DE LA COMPUTACIÓN E INTELIGENCIA ARTIFICIAL

UNIVERSIDAD DE GRANADA

Curso 2015-2016

La solución para este sistema con un comportamiento totalmente deliberativo consta de:

1. **PODA ALFA-BETA**

Como primer objetivo principal de esta práctica es la implementación del algoritmo PODA ALFA-BETA, el que se encarga de crear un árbol de posibles acciones, en este caso con profundidad máxima de 8, y elegir entre todas las posibles, la mejor.

Para ello implemento el método "Poda_AlfaBeta ()", que recibe 7 parámetros: el tablero actual que se encuentra la partida, el número de jugador que desempeñamos en la partida ("si soy el jugador 1, o el jugador 2"), el nivel de profundidad que no es encontramos ("en este caso siempre será 0, ya que cada vez que realizamos la llamada a este método calcula el árbol completo desde la raíz, es decir, desde la jugada que nos encontramos"), la profundidad máxima que va a llegar a ramificar el árbol ("siempre será 8, ya que tenemos ahí el límite), la acción que realizaremos dependiendo de la mejor jugada encontrada, y los valores de alfa, y beta que siempre serán menos infinito para alfa y más infinito para beta("el máximo valor que puede tomar la variable de tipo double"), respectivamente.

Resumiendo, esta función creará el árbol de soluciones recursivamente, es decir, desde el nodo padre se irá llamando a esta misma función para todos sus hijos, y así sucesivamente a los hijos de los hijos, hasta llegar a un nodo terminal o el límite de profundidad. Nos tenemos que ayudar mediante las posibles acciones, es decir, calculamos las acciones que tenemos disponibles mediante la función "posible_actions ()" ya implementada, por ejemplo, si podemos poner una ficha en la columna 1, columna2…, columna 7, y si tenemos disponible la ficha bomba para la acción BOOM.

A través de esto anterior podemos crear el árbol de jugadas, por ejemplo, si escogemos poner una ficha en la columna 1 ("Si está disponible esta acción"), ya podemos generar un nivel de profundidad. Un ejemplo: analizamos que ocurriría si hemos elegido poner una ficha en la columna 1 y la siguiente jugada ponemos también una ficha en la columna 1 "se quedaría dos fichas en la misma columna", o en la columna 2 "se quedarían dos fichas en la misma fila… Estas situaciones tenemos que evaluarlas y darle una valoración determinada, para ello hablaremos en el punto 2.

Tenemos que tener en cuenta que debemos de diferenciar entre los nodos MAX y los nodos MIN, ya me ha faltado mencionar anteriormente que este algoritmo realiza la función del algoritmo MINMAX añadiendo la poda Alfa-Beta. Si el nodo es par estamos hablando de un nodo MAX y si es impar hablamos de un nodo MIN. El nodo MAX será encargado de maximizar sus hijos (el máximo valor de todos sus hijos) y devuelve el valor de alfa, por el contrario, el nodo MIN minimiza sus hijos (el mínimo valor de todos sus hijos) y devuelve el valor de beta. Así sucesivamente arrastrando la mejor jugada hasta llegar al nodo MAX de la raíz del árbol.

Por último, añadimos "la poda" que solo debemos de respetar la condición de parada, es decir, en cada nodo, sea MAX o MIN, si se cumple la condición de que el valor de beta es mayor que el valor de alfa, se realiza la poda, por lo que no debemos de seguir explorando lo demás hijos de esa rama del árbol, o descartar esa rama.

2. HEURISTICA

Este apartado es el encargado de valorar un posible tablero generado, “hay que penalizar las situaciones no factibles para nosotros” para que el método Poda_AlfaBeta () evite estas situaciones.

Para ello, implemento el método “Valoracion ()” que es llamado por cada tablero que genera el método anterior Poda_AlfaBeta () y se le asigna un valor.

Para ello, en primer lugar, compruebo que no es un tablero ganador, empate o pérdida de la partida para el jugador, si no se cumple ninguna de estás condiciones anteriores recorro el tablero como una matriz de 7x7, dando valores según las distintas jugadas.

Ya en segundo lugar aprovecho la función “Get_Ocupacion_Columna ()” ya implementada que me devuelve si la columna especificada por parámetro está completa o tiene casillas libres, para evitar recorrer columnas completas y hacer un poco más eficiente esta función.

Por último, en este sistema de juego podemos realizar el papel de jugador 1, el que comienza la partida, o jugador 2, que es el jugador que continua la partida después del jugador 1. Indico esto porque dependiendo del jugador que seamos en la partida, debe de tener en cuenta las fichas de un jugador u otro.

Teniendo en cuenta las condiciones anteriores, resumiendo un poco, voy valorando el tablero actual de la siguiente manera: (“voy a considerar que soy el jugador 1, ya que si es el jugador 2 sería la misma valoración, pero con las fichas contrarias”):

1 Aumento la valoración positivamente si, en la posición “i,j” de la matriz, a la derecha, izquierda o justo encima hay una ficha del jugador contrario.
2 Y a partir de ahora ya sólo disminuyo la valoración:
  2.2 En menor medida (intervalos negativos aproximados a 0):
    - Si hay una ficha bomba del jugador contrario justo encima.
    - Si tengo dos fichas juntas en posición horizontal
    - Si tengo dos fichas juntas en posición vertical
    - Si tengo dos fichas juntas en diagonal (“en forma de cruz, es decir, hacia la izquierda-arriba, derecha-arriba, izquierda-abajo, derecha-abajo”).
  2.3 Para las mismas situaciones anteriores del punto 2.2 y con una valoración mayor negativamente, pero en este caso, con 3 fichas consecutivas (intervalos negativos más alejados de 0);
  2.4 Tengo en cuenta también las posibles jugadas del punto 2.3 que serían 2 fichas consecutivas, con “un hueco en medio” y la cuarto ficha que formaría “en 4 en raya” con la misma valoración negativa.

Una vez calculado la valoración completa de toda la matriz (“tablero”) la función devuelve dicho valor para ese tablero.

MARCOS AVILÉS LUQUE
Grupo: 8
Práctica 3