José Santos Salvador

Memoria práctica 3 Inteligencia Artificial

1. Análisis del problema

   La práctica consiste en desarrollar un agente deliberativo capaz de ganar contra su oponente en el juego de Desconecta-4 BOOM. Para ello es necesario apoyarnos sobre la teoría de Juegos bipersonales con información perfecta. En los juegos cada uno de los participantes intenta conseguir el mayor beneficio para ganar, una solución sería aquella que permite alcanzar la victoria o qué resultado se puede esperar.

   El objetivo del juego es hacer alinear las fichas del rival, ya sea en horizontal, vertical o en diagonal. Para eliminar la posibilidad de empate se crea una ficha especial llamada bomba que elimina las fichas que se encuentren en la misma fila que la susodicha y desplazaría el resto de las que están encima hacia abajo.

   Para resolver este juego es necesario hacer uso de un árbol de exploración de juegos(representación explícita de todas las formas de jugar a un juego). Se puede hacer uso del algoritmo min-max o de la poda alfa-beta(se obtiene el mismo resultado que con el min-max pero con menos esfuerzo computacional o en el peor caso el mismo).

   La poda alfa beta es una técnica de búsqueda que mejora el algoritmo minmax ya que reduce el numero de nodos a evaluar. En el árbol cada uno de sus hijos son las posibles jugadas hasta llegar a los nodos hoja, estos nodos se evalúan con una heurística y se va añadiendo el valor del hijo que cumpla los requisitos del nivel(nodo de minimizar o maximizar) al padre. En caso de que beta sea menor o igual a alfa se poda.

2. Descripción de la solución planteada

   Para implementar la poda alfa-beta es necesario modificar el método think que invoca al siguiente movimiento del jugador, crear la función poda alfa-beta y la función valoración que es la heurística que nos permite valorar cada jugada para su elección o poda.

   - En la **función think** llamamos a Poda_AlfaBeta con el estado del tablero actual, el jugador que está jugando, la profundidad máxima permitida para la este algoritmo, la variable accion que nos devolvera la accion a realizar, los valores de alfa y beta, que en este caso son menosinf y masinf respectivamente y el numero de nodos(inicialmente en 0),la función devuelve el valor de Alfabeta. Mostramos por pantalla el valor junto con la accion a realizar y se hace un return de la accion por parte del think.
     También contemplo en el think si el jugador tiene una bomba y puede ganar la partida haciéndola explotar la acción que devuelve el método es BOOM, para ello uso Have_BOOM() que me dice si el jugador tiene bomba, después

utilizo un tablero auxiliar al que le aplico la bomba y por último llamo a RevisarTablero() para ver si gana así o no.

- En la **función Poda_AlfaBeta** si la profundidad es cero (significa que hemos recorrido desde PROFUNDIDAD_ALFABETA hasta 0) o la llamada al tablero sobre el final del juego(E.JuegoTerminado()) , la función suma uno más al número de nodos y devuelve el valor del AlfaBeta según dicho estado de juego y el jugador que está jugando(Valoración(E,jugador)).

  Después dependiendo de si la variable max_min está a verdadero o falso(maximizar o minimizar) se realiza una llamada por cada hijo(posibles acciones) con la siguiente acción a la función Poda_AlfaBeta al contrario de la variable max_min y con profundidad-1. Si el valor que devuelve el hijo es mayor que alpha, se actualiza alpha a ese valor y la accion, en cambio si beta es menor o igual que alpha se sale.

- La **función Valoracion** que es llamada por la Poda_AlfaBeta devuelve el valor heurístico según el jugador que está jugando y el tablero actual.
  Primero revisa el tablero por si has ganado o perdido y devuelve el valor de masinf o menosinf(si gana o pierde respectivamente).
  En caso de no haber perdido o ganado, se hace una llamada a una función que he llamado Heuristica, si eres el jugador 1 se resta el valor que devuelva tu Heuristica al valor de la Heuristica del jugador 2, en caso de ser el jugador 2 el procedimiento es análogo.

- La **función Heurística** recorre el tablero y le pasa unos índices a las funciones Columna, Filas y Diagonal sobre los que calcular el valor según la cantidad de fichas que hay repetidas de forma seguida(en las distintas formas), todo ello lo hago con un valor positivo y en el momento de devolver el valor de la función Heurística lo devuelvo el negativo ya que el objetivo es hacer que el oponente haga 4 fichas seguidas y por tanto mientras más tengas tu seguidas más posibilidades tienes de perder.

  La **función Columna** tras pasarle la columna recorre la matriz por filas y va mirando si la casilla es del jugador que estamos mirando y si lo es suma a la variable repetidas uno, tras acabar con el recorrido, según la cantidad de repetidas que hay le asigno un valor desde 10 hasta 100(siendo esta la máxima cantidad de fichas seguidas posibles que es 3).

  La **función Filas** realiza lo mismo que la de Columna pero en vez de recorrerla por filas es por columnas y le pasas la fila a la función.

  La **función Diagonal** realiza lo mismo que Columna y FIlas pero la forma de recorrer es diferente ya que recorre por filas y le pasas por cabecera la columna pero miras según si estás en la diagonal de la parte derecha(fil<4 && col<4) y si es así entonces la forma de mirar si la casilla es de jugador es

sumandole a las filas y las columnas un contador que va incrementando. Si en cambio es la diagonal de la parte izquierda(fil>2 && col<4) vas mirando la casilla restándole en las filas y sumandole en las columnas con dicho contador.

En el transcurso de la práctica he ido cambiando los valores que daba según la cantidad de fichas repetidas, al inicio daba valores muy altos y de forma aleatoria y dando valores si se encontraba la bomba tuya y la del enemigo, sin embargo conseguía ajustar la heurística de forma que me funcionase con alguna mejora sustancial respecto a la ya elegida, entonces decidí dejar esta heurística que aunque es simple, funciona y acaba ganando a todos los ninjas con las distintas posibilidades.