Inteligencia Artificial - Jose Luis Gallego Peña (Grupo A2)

Documentación de la práctica 3: Métodos de Búsqueda con Adversario (Juegos)

Desconecta-4 BOOM

Análisis del problema

El problema consiste en resolver Desconecta-4 BOOM, un juego en el que un jugador se enfrenta a otro en un tablero de 7x7 casillas. Cada jugador debe introducir, en su turno, una ficha de su color en una de las 7 casillas. Un jugador gana en el momento en el que su oponente tiene 4 fichas consecutivas en el tablero de forma horizontal, vertical o diagonal, por lo que el jugador deberá intentar en todo momento que sus fichas no junten 4 y además forzar al oponente a juntar sus fichas para así ganar. Sin embargo, cada 5 turnos del jugador, la ficha que ponga será una ficha con bomba, que permitirá al jugador elegir quitarse todas las fichas de la línea en la que esté explotando la bomba.

Desconecta-4 BOOM es un juego bipersonal con información perfecta. El ser bipersonal significa que juegan dos jugadores, el uno contra el otro; y el ser con información perfecta significa que se conoce en todo momento el estado del juego y lo que hace el oponente (ambos jugadores ven lo que pasa en el tablero, qué ficha pone uno mismo y qué ficha pone el oponente). Puesto que es competitivo, un jugador compite contra el otro, intentando conseguir el mayor beneficio para sus intereses. En la situación final del juego solo hay tres posibilidades: ganar, perder, o empatar; el beneficio de un jugador (que gane) significa la total pérdida de beneficio del otro (que pierda), por lo que podemos afirmar que este juego es un juego de suma nula.

Para que el bot pueda jugar alcanzando el máximo beneficio (ganar) debemos enfocar el juego como un problema de búsqueda con las siguientes características:

- **Estado inicial**: Tablero 7x7 donde el jugador 1 de color verde le toca poner ficha en una de las 7 columnas.
- **Función sucesor**: Lista de entre 1 y 8 pares de (movimiento, estado) según estén disponibles, donde *movimiento* es cada uno de los movimientos legales disponibles en ese momento de la partida: poner ficha en columna 1, 2, 3, 4, 5, 6, 7 o explotar la bomba, y *estado* es el estado resultante de ese movimiento, es decir, el tablero con sus fichas una vez que se ha realizado la acción.
- **Test terminal**: El juego termina cuando uno de los dos jugadores tiene 4 fichas consecutivas de forma horizontal, vertical o diagonal.
- **Función de valoración**: Victoria: $+\infty$ (+INFINITO), Derrota: $-\infty$ (-INFINITO), Empate: 0.

Teniendo en cuenta esto, se usará un árbol de exploración de juegos, que nos dará una representación explícita de todas las formas de jugar la partida del juego y así poder establecer mediante un algoritmo el ganar o perder un juego a partir de una situación inicial dada.

Descripción de la solución planteada

Técnica de búsqueda con adversario: Poda Alfa-Beta

La técnica de búsqueda con adversario a utilizar será el algoritmo Minimax con Poda Alfa-Beta con profundidad limitada y cota máxima de 8, para así dotar de comportamiento inteligente deliberativo al jugador artificial del juego y poder ganar en él.

Este algoritmo usa la notación MIN-MAX, para la cual tenemos un arbol de estados con nodos MAX y nodos MIN que se van alternando, en el que MAX intentará maximizar su beneficio a la vez que intenta minimizar el de MIN, y viceversa. Por tanto, tenemos que tanto MAX como MIN son jugadores, perteneciendo en este caso los nodos MAX al jugador que lleve este algoritmo y los nodos MIN al oponente (los ninjas de la práctica). Sabiendo esto, vemos que MIN tiene algo que decir al respecto de las decisiones que se toman a lo

largo del árbol, teniendo como solución una secuencia de movimientos que llevan a un estado terminal, por lo que MAX encontrará una estrategia que tenga en cuenta los movimientos de MIN.

Mediante la Poda Alfa-Beta se obtiene el mismo resultado que con el algoritmo MiniMax normal pero con una mejor eficiencia. En este algoritmo cada nodo tiene asociado dos variables: alfa y beta. Alfa representa el mejor valor encontrado hasta el momento por MAX, y beta representa el mejor valor encontrado hasta el momento por los nodos MIN. La implementación del algoritmo es una función recursiva que recorre todo el árbol generando la valoración para todos sus nodos y la próxima acción a realizar más favorable a partir del estado actual. La función recibe el estado del juego actual, la profundidad máxima, los valores de alfa y beta, el jugador que llama al algoritmo y por referencia la acción que se va a realizar una vez que se exploren los nodos. Funciona de la siguiente manera:

- Mediante la función GenerateAllMoves se genera un vector con todos los hijos posibles del nodo actual.
- En el caso base se calcula la valoración heurística del nodo cuando se ha alcanzado el nodo terminal, para así luego ir actualizando el valor de los nodos conforme vuelve la recursividad al estado inicial. Se sabe que se está en un nodo terminal ya que en cada llamada recursiva se decrementa la profundidad máxima de 8 en 1, por lo que cuando llegue a 0 será la profundidad máxima.
- Para la exploración de los nodos se distingue entre nodos MIN y nodos MAX mediante la profundidad: si es par será un nodo MAX, ya que la profundidad máxima es 8 (número par) y el primer estado del árbol siempre es MAX, y si es impar será un nodo MIN. En ambos casos se cumple el criterio de poda cuando alfa (que debe crecer), supera a beta (que debe crecer), es decir, se cruzan los valores. Para cada uno de los nodos hijos generados:
  - MAX: Se calcula el valor minimax de cada hijo llamando recursivamente a la función (baja la recursividad). Si este valor es mayor que el valor de alfa del nodo actual, se actualiza el valor de alfa ya que MAX busca que la cota inferior alfa crezca. En este caso, puesto que los nodos MAX serán siempre los de mi jugador se actualiza la acción que se pasó por referencia una vez que estoy en el nodo raíz, ya que la acción a realizar es mejor que la que tenía, y para saber si está en el nodo raíz se compara la profundidad actual con la profundidad máxima, indicando que es un nodo raíz puesto que es donde se empieza a decrementar; esto mejora mucho la eficiencia del algoritmo, ya que en un primer lugar se intentó comparar la variable *estado* pasada como parámetro y el atributo *actual_* de la clase, es decir, comparar nodos, sin embargo esta comparación era bastante exigente en eficiencia, La acción a realizar en cada momento de la partida es uno de las 8 acciones hijas del estado actual.
    Por último, si se cumple el criterio de poda, el valor actual que se está examinando es peor, y por tanto se deja de explorar el estado terminando la función devolviendo el valor del contrario: beta.
  - MIN: Se calcula el valor minimax de cada hijo llamando recursivamente a la función (baja la recursividad). Si este valor es menor que el valor de beta del nodo actual, se actualiza el valor de beta ya que MIN busca que la cota superior beta decrezca. Si se cumple el criterio de poda, el valor actual que se está examinando es peor, y por tanto se deja de explorar el estado terminando la función devolviendo el valor del contrario: alfa. En este caso no se calcula la acción puesto que no es necesaria para MAX, solo necesita el valor minimax.

Función heurística

La función heurística es la parte principal del algoritmo y es la que determina cómo se obtiene el beneficio y cómo ganar. Esta heurística consiste esencialmente en un valor absoluto de lo cerca que está el jugador de ganar - lo cerca que está el oponente de ganar. En este caso, lo cerca que está el jugador de ganar significa que el oponente tenga muchas fichas, y lo cerca que está de perder significa que tenga muchas fichas, por tanto la idea central de la heurística es que cuantas menos fichas tenga el jugador y más el oponente mejor, puesto que de esa forma más oportunidades habrá de que el oponente ponga cuatro fichas en línea y así ganar.

Se realizan cuatro recorridos a la matriz estado, una para cada forma en la que pueden estar juntas las fichas: en horizontal, vertical, diagonal creciente y diagonal decreciente. En cada uno de los recorridos se cuenta

el número de fichas consecutivas de cada jugador, y se proporciona un valor positivo o negativo según se ha especificado anteriormente. Sin embargo, este valor es más positivo o más negativo teniendo en cuenta cuántas fichas consecutivas hay: para el jugador tener 3 fichas consecutivas es más negativo que tener solo 1 o 2 consecutivas, y al revés, que el oponente tenga 3 fichas consecutivas es mucho más favorable que si tuviese 1 o 2. Estos valores son de 10, 20 o 30, positivos o negativos, en función de si se tienen 1, 2 o 3 fichas consecutivas.

Además, se controlan los estados en los que hay 4 fichas consecutivas, si las hay se añade a la valoración un número muy grande en comparación a los demás (9999), para así evitar este estado (si son las fichas del jugador) o buscarlo (si son las fichas del oponente).

Por último queda comentar la importancia de la bomba en esta heurística. Puesto que tenemos muchas situaciones en las que explotando la bomba las fichas del oponente bajan y se conectan 4, se ha optado por darle una prioridad especial a la bomba. De esta manera, mediante la funicón Have_BOOM se le ha dado un plus de valoración al estado si este tiene una bomba, para así explotar de forma más frecuente. Además, para maximizar esto, en la función de poda alfabeta, para los nodos MAX en concreto se recorren los hijos primero desde la bomba (es decir, el vector de estados desde el final) para así, en caso de empate, quedarse simplemente con el primero que se ha escogido, es decir, la bomba.

Jugando contra los 3 ninjas se les ha ganado a los tres, siendo tanto jugador 1 como jugador 2, y se han obtenido los siguientes resultados:

- **Heurística VS Ninja 1:**
  - Siendo jugador 1 (Verde): 41 jugadas
  - Siendo jugador 2 (Azul): 55 jugadas
- **Heurística VS Ninja 2:**
  - Siendo jugador 1 (Verde): 31 jugadas
  - Siendo jugador 2 (Azul): 30 jugadas
- **Heurística VS Ninja 3:**
  - Siendo jugador 1 (Verde): 26 jugadas
  - Siendo jugador 2 (Azul): 22 jugadas

Por tanto al final nos queda que la valoración heurística es un valor al que se le va sumando o restando conforme la situación que se tenga, quedando al final un valor más positivo o más negativo, con el que se determinará qué nodo escogerá el algoritmo AlfaBeta y que, a parte de por ganar, podemos ver que la heurística realiza bien su función ya que cuando termina la partida (y en general, en la gran mayoría de los estados del juego) el jugador tiene menos fichas que el oponente, que es lo que se busca.