UNIVERSIDAD DE GRANADA

ETSIIT

PRÁCTICA 3:

El Parchís (BOOM!)

BÚSQUEDA CON ADVERSARIO (JUEGOS)

Isabel Morro Tabares
79095945A
Inteligencia Artificial
A2D, DGIIM

Índice

Análisis del problema

Esta tercera práctica de la asignatura Inteligencia Artificial consiste en el diseño e implementación de alguna de las técnicas de búsqueda con adversario en un entorno de juegos. En nuestro caso, trabajamos sobre el juego Parchís con aspectos añadidos del juego Mario Bros. En este, se enfrentan dos jugadores, cada uno de ellos con dos colores, y gana aquel que consiga meter todas las fichas de uno de sus colores en la meta.

En lo que se refiere al análisis del problema por parte del alumno, se debe dotar de comportamiento inteligente deliberativo a nuestro "jugador virtual" de forma que calcule sus mejores movimientos, estime los posibles daños por parte del adversario y gane la partida. Para ello, aparte de la implementación de la técnica de búsqueda, se tiene como principal objetivo hallar una heurística que nos permita ganar las partidas a los ninjas proporcionados.

En mi caso, el algoritmo escogido ha sido Poda Alfa-Beta, pues este trabaja de igual forma que MiniMax pero descartando aquellos caminos en los que se garantiza que no se llegará a un estado óptimo, lo que hace que sea un algoritmo más eficiente, por tanto nos proporciona una búsqueda más rápida y un menor tiempo de espera para mover cada ficha.

Solución planteada

¿Cómo trabaja el algoritmo Poda Alfa-Beta?

Nuestro objetivo principal es buscar la ganancia máxima para nuestro jugador, por ello en los nodos donde el jugador que mueva seamos nosotros, buscamos el valor máximo, al contrario que cuando juega el oponente, donde buscamos la ganancia mínima.

El algoritmo trabaja sobre el valor de alpha y beta, el estado, el jugador y la profundidad actual del juego, la profundidad máxima permitida y, el color, la pieza y el dado a usar.

Trabajamos con búsqueda de hasta profundidad 6. Por ello, en primer lugar comprobamos si el nodo ha alcanzado la profundidad máxima o el juego ha acabado, en cuyo caso, devolvemos el valor de la heurística. En caso contrario, seguimos ampliando nuestro árbol de búsqueda investigando las posibles jugadas de forma que se maximice alpha (para jugadores) o se minimice beta (para oponentes).

Para cada nodo, obtenemos un iterador que recorre sus nodos hijo, y a cada uno de estos le aplicamos el algoritmo Poda Alfa-Beta por recurrencia. Ahora bien, en el caso de los jugadores, si el valor obtenido para dicho nodo es mayor que los obtenidos anteriormente, se toma dicho valor como el nuevo alpha. Además, si el nuevo alpha es mayor que beta, significa que el nodo actual no podrá mejorar el alpha ya encontrado, por ello, realizamos la poda.

En el caso de ser nodo mínimo, actualizamos beta si obtenemos un valor menor que el actual, y volvemos a comprobar si alpha es mayor o igual que beta.

Finalmente, cuando regresamos al nodo raíz, actualizamos los valores del color, la pieza y el dado que se ha estimado que nos llevarán a un mejor movimiento para nosotros, y un peor movimiento para nuestro contrincante.

Heurísticas empleadas

En mi caso, se pueden encontrar en el código dos heurísticas: *MiValoración1* y *MiValoración2*; sin embargo, *MiValoración2* es una ampliación de *MiValoración1*, por lo tanto trataré esta heurística.

Mejor Heurística

La heurística final que he escogido es *MiValoracion2*, por razones anteriormente comentadas.

El método *MiValoración2* devuelve más infinito si ganamos y menos infinito si gana nuestro oponente. En caso contrario, se calcula la puntuación que puede llegar a obtener nuestro jugador y se le resta la puntuación que podría alcanzar nuestro contrincante.

Las puntuaciones de ambos se calculan de la misma forma, por ello, para obtenerlas se llama al método *calculosJugadores2* que recibe el estado del juego y el jugador actual. Pero, ¿en qué se basa *calculoJugadores2*?

Principalmente, dependiendo de lo que ocurra en el estado actual de la partida, el jugador suma puntos según los beneficios que puede llegar a obtener o resta según las desventajas que pueden aparecer. Para dicho jugador, examinamos sus dos colores y sus 3 fichas de cada uno de dichos colores. Doy prioridad a algunos movimientos o estados frente a otros sumando un valor mayor. Enumero a continuación dichos valores y prioridades, diferenciando entre puntuaciones constantes y no constantes:

- Puntuaciones constantes:

1. La puntuación más alta se obtiene si se trata de un movimiento donde se come a otra ficha, con una puntuación de 250.
2. Si el movimiento nos lleva a la victoria, incrementa la puntuación en 15.
3. Si en dicho estado existen movimientos de dado especial que no nos perjudican, se aumenta la puntuación más o menos, dependiendo del número de casillas que se recorre.
4. Si las fichas están en una casilla segura, se aumenta la puntuación.
5. Si hay fichas sin jugar, es decir, en “casa”, se resta un valor de tres por ficha en casa.
6. Si hay fichas en la meta, se incrementa la puntuación en uno por cada una de ellas.
7. Se beneficia con un valor de 2 si las fichas del jugador han formado barrera.
8. Por cada ficha colocada delante de una ficha del oponente a menos de 8 casillas (sin contar el 3, pues no hay dado de valor tres), se decrementa en 1 la puntuación, pues la probabilidad de que nos coman una ficha es mayor.

9. Se suma puntuación si las fichas destruidas en el último movimiento son las de los colores de nuestro contrincante.

- Puntuaciones no constantes

10. Se beneficia el estado que tenga las fichas de jugador más cerca de la meta. Se incrementa la puntuación con el resultado de la diferencia del número de casillas normales del tablero menos la distancia de la ficha a meta.
11. Si tenemos ya dos fichas de un mismo color en la meta y la tercera ficha de ese mismo color no está en "casa", se beneficia con la puntuación más alta de la heurística. Esto es, cuanto más cerca esté esa tercera ficha de la meta, más se sumará. Intentando así que tan solo se mueva la ficha que falta de aquel color que nos hará ganar, evitando mover el otro color.

Cabe resaltar que, tanto los casos catastróficos por usar el dado especial con un nivel de energía que nos perjudica como el hecho de comernos una ficha de nuestros propios colores, aunque no están especificados en la heurística, están controlados. Pues, con el cálculo de las distancias de las fichas a meta y de las fichas en "casa", nunca será favorable un estado donde provoquemos la pérdida de nuestras propias fichas.

Ventajas e Inconvenientes

La heurística planteada gana al ninja 1 y al ninja 2 (siendo jugador 1 o jugador 2), y al ninja 3 siendo jugador 1.

Plantea diversas ventajas como: la prioridad al movimiento de comer fichas (no solo perjudica al oponente, sino que nos permite avanzar más rápido), el buen uso del dado especial, la capacidad de mover aquellas fichas que se encuentran más cerca de la meta aumentando así la probabilidad de llegar a la meta lo antes posible e intentar evitar que el oponente nos coma con el uso del dado normal, posicionandonos en aquellas casillas inalcanzables para este.

Como principal desventaja se puede destacar que, a pesar de estar implementado en el código el punto 11 de la enumeración anterior, esta heurística no termina de mover solo la ficha que nos queda para ganar la partida, si no que sigue moviendo el resto de fichas aunque no le lleven a ganar la partida de forma más rápida (caso del ninja 3 siendo jugador 2).