MEMORIA PARCHÍS

Inteligencia Artificial - Práctica 3

UNIVERSIDAD DE GRANADA

Laura Lázaro Soraluce

2023/2024

PODA

Para la poda, me he basado en el siguiente pseudocódigo [1]:

```
función alfa-beta(nodo //en nuestro caso el tablero, profundidad,
α, β, jugador)
    si nodo es un nodo terminal o profundidad = 0
        devolver el valor heurístico del nodo
    si jugador1
        para cada hijo de nodo
            α := max(α, alfa-beta(hijo, profundidad-1, α, β,
jugador2))
            si β≤α
                romper (* poda β *)
        devolver α
    si no
        para cada hijo de nodo
            β := min(β, alfa-beta(hijo, profundidad-1, α, β,
jugador1))
            si β≤α
                romper (* poda α *)
        devolver β
```

```
(* Llamada inicial *)
alfa-beta(origen, profundidad, -infinito, +infinito,
jugador_deseado)
```

Primero cojo todos los hijos posibles, es decir, todos los estados de la partida a los que puedo llegar desde el estado actual con un único movimiento. Siempre que la partida no haya acabado y no haya llegado a la profundidad máxima (que en este caso es 6), recorro dichos hijos. Para cada hijo, se van a hacer llamadas recursivas al método, a no ser que la profundidad sea el tope, en cuyo caso se devuelve el valor de la heurística para el estado actual, que comentaremos más adelante. Si estamos en un nodo máx, es decir, le toca a mi jugador, la heurística que nos va a devolver las llamadas recursivas al método va a ser un posible alfa. Lo comparamos con el alfa que ya tenemos; si el alfa nuevo es mayor, nos lo quedamos; si no, lo ignoramos. Esto es porque los nodos máx actualizan su nodo alfa con los valores que les vienen de los nodos hijos. Asimismo, estos nodos van a devolver el valor de alfa, pues los nodos máx siempre pasan este valor hacia arriba. En el caso de los nodos min, pasa al contrario: actualizamos con el posible beta si este es mayor que el beta que ya había, y es también el beta el que devolvemos, pasándolo hacia arriba. En ambos casos (nodo min y nodo max), se para de buscar en los próximos hijos si en algún momento se cumple la condición de poda: beta<=alpha.

HEURÍSTICA

Mi heurística la he hecho de manera simétrica, es decir, la manera en la que sumo/resto puntos a mi jugador, es la misma en la que sumo/resto puntos al jugador contrario al analizar el estado del tablero. Hay 5 cosas que he tenido en cuenta (Tipos de casillas, Dados, Distancia al objetivo, Si he comido, Si me han comido):

- Para cada pieza de cada color, según el **tipo de casilla en el que se encuentra**:
    - Si la pieza está en una casilla segura: +5.
    - Si la pieza está en el objetivo: +20.
    - Si la pieza está en la cola final: +10. Menor que cuando está en el objetivo para que no se quede en la cola final mucho tiempo, sino que intente llegar al objetivo cuanto antes.
    - Si la pieza está en casa: -80. Esto lo hago para estresar lo malo que es estar en casa, pues obliga a esperar a tener un dado 5 para gastarlo en ella.
- Para cada pieza de cada color, según **si han sido comidas**:
    - Si la última pieza comida en la partida coincide con la que estoy mirando (lo compruebo con la función eatenPiece()): -10. Esto lo hago porque, independientemente de si ha sido mi propio jugador o no, quien se ha comido mi ficha, interpreto que es preferible que no me la coman.
- Según la **distancia a que se encuentren del objetivo**: Utilizo un array distancia[2] de enteros, para guardar la distancia total a la que están las fichas de cada uno de mis colores. Es decir, distancia[0] guardará la suma de las distancias de todas las fichas de mi color 0, y distancia[1] lo mismo pero con mi color 1. Esto lo utilizo para saber cuál de mis dos colores está más adelantado.
    - Para cada pieza de cada color, resto la cantidad de casillas que le queden hasta el objetivo.
    - Para cada pieza de mi color más avanzado, resto de nuevo la cantidad de casillas que le queden hasta el objetivo (es como si las restara x2 para este color).
- Para cada pieza de mi color más avanzado (análogo para el color más avanzado del oponente) según en qué **tipo de casilla se encuentra**:
    - Si la pieza está en el objetivo: +120.
    - Si la pieza está en la cola final, +100. Utilizo estas cantidades tan elevadas, para que, una vez uno de mis dos colores esté claramente más avanzado que el otro, sea él quien intente llevar todas sus piezas al objetivo. Intento

con esto evitar perder tiempo llevando fichas de mi color menos avanzado a la meta, ya que prefiero que simplemente se queden estorbando al otro jugador.

- Si la **última ficha comida** es una de mi color más avanzado: -15. Esto lo hago para enfatizar que es peor que el oponente (o mi propio jugador) se coma una ficha de mi color más avanzado. Puestos a comer, prefiero que me coma una del color menos avanzado. Con esto también intento evitar que mi color menos avanzado se coma al otro.
- Según los **dados que tenga**. El orden de menor valorado a mayor valorado es el siguiente: banana=mushroom < boo < red_shell=shock < horn < blue_shell < mega_mushroom < bullet < star. Este orden lo he decidido por preferencias personales, según lo que considero más útil.
- Según **a quién me he comido**: si tengo un dado +20 (me he comido a alguien) y la última ficha comida es del color más avanzado del jugador contrario, sumo 20 a mi puntuación. Esto es porque considero para promover que mi jugador se coma a las fichas más avanzadas del contrincante, y no a una ficha cualquiera.

REFERENCIAS

1. colaboradores de Wikipedia. (2023). Poda alfa-beta. *Wikipedia, la enciclopedia libre*. https://es.wikipedia.org/wiki/Poda_alfa-beta