Práctica 3:
Búsqueda con Adversario (Juegos)

El Parchís

Linqi Zhu
IA 2_D subgrupo D1
10 de junio de 2022

UNIVERSIDAD DE GRANADA

1. Análisis del problema

Desarrollar un programa (modificando el código de los ficheros del simulador **AIPlayer.cpp, AIPlayer.h**) que implemente el algoritmo MINMAX o la PODA ALFA-BETA en los término que se ha explicado en el guión de la práctica, de manera que un jugador pueda determinar el movimiento más prometedor para ganar el jugo, explorando el árbol de juego desde el estado actual hasta una profundidad máxima de 6 dada como entrada al algoritmo.

También hay definir una heurística apropiada, que asociada al algoritmo mencionado anteriormente proporcione un buen jugador artificial para el juego del PARCHÍS.

Diseño del algoritmo PODA ALFA-BETA :

```
function poda-alfa-beta(profundidad, alpha, beta):
   int mejor_hasta_momento;
   if profundidad = profundidad_max o final_del_juego:
      return valor_heuristivco;
   else
      if player = MAX :
         while (!(next_player == actual)) // Para cada hijo del nodo
            mejor_hasta_momento = poda-alfa-beta(profundidad + 1,α,β)
               if mejor_hasta_momento > α
                  α = mejor_hasta_momento
                  if α >= β: podar
         end
      return α
      end
      if player = MIN :
         while (!(next_player == actual)) // Para cada hijo del nodo
            mejor_hasta_momento = poda-alfa-beta(profundidad + 1,α,β)
               if mejor_hasta_momento > α
                  α = mejor_hasta_momento
               if α >= β: podar
          end
      return β
      end
    end
```

Se utiliza este algoritmo y no el minmax, porque reduce el número de nodos evaluados en el árbol del juego. Para ello, trata de eliminar partes grandes del árbol que se va construyendo de forma que se devuelve el mismo movimiento que devolverá este, podando ramas que se sepa que no van influir la decisión final.

2. La Heurística implementado

```
static double MiValoracion1(const Parchis &estado, int jugador);
```

En comparación con la heurística Valoración Test hay más criterio para puntuación.

-Valoro positivamente las fichas de los colores de mi jugador que están en la casilla segura, la meta o cola final con diferentes puntuación, según el nivel de seguridad de las fichas, más puntuación o menos. También en caso de que la última acción realizada por dicha ficha sea de comer a otro ficha, ya que pueden avanzar más paso que tirando los dados. Y se penaliza por las fichas del oponente que estén en la misma situación.

- Valoro negativamente las fichas de los colores de mi jugador que están en la casilla home sin salir. Ya que parar ganar el juego es necesario que todas las ficha salgan del home. También en el caso de no llegue a la meta, la puntuación será el número de pasos que hay que avanzar para llegar a la meta. Y se despenaliza por las fichas del oponente que estén en la misma situación.

He utilizado esta heurística para mi algoritmo, ya que he conseguido ganar a los ninjas, las ventajas que tiene frente a otro (ValoracionTest o MiValoracion2) es que tiene en cuenta más detalles. Y las desventajas al tener más detalles el algoritmo tardan más en ejecución pero a ser poco lo podemos descartar.