Práctica 3 Inteligencia Artificial: Agentes en Juegos

Julio José Reyes Hurtado

June 3, 2018

1 Introducción

En el presente documento vamos a realizar una explicación sobre el bot creado para jugar al juego de Mancala. Se detallarán aspectos como las estructuras de datos utilizadas, lugar donde se encuentran estas estructuras de datos, algoritmo de resolución que se ha usado, explicando el diseño ejecutado así como la heurística utilizada.

2 Diseño de estado

Para la realización de este trabajo el árbol de estados que se genera está constituido por un struct nodo.

Este struct se situa dentro del fichero boTijo.h y está compuesto por un estado de la clase GamesState con todo su contenido, un movimiento del tipo Move y una heurística.

Gracias a este diseño del árbol de estados podremos almacenar la información relativa a los distintos estados que se generan desde el inicial, pudiendo así en la función NextMove devolver el movimiento almacenado en el nodo que contiene el siguiente movimiento.

3 Descripción del algoritmo implementado

Para la realización de esta práctica se ha utilizado una variante del algorimo minimax con poda alfa-beta. En lugar de devolver la heuristica del mejor movimiento siguiente con nuestra implementación de nodos el diseño del alfa-beta cambia a solo utilizar esta estructura de datos no teniendo así que trabajar con movimientos ni enteros, usando así tanto para el siguiente movimiento como para alfa y beta los nodos explicados en la sección anterior que contienen toda la información necesaria referente a los datos que es necesario utilizar en las operaciones realizadas en el algoritmo.

Para la resolución en la función NextMove vamos a inicializar el nodo inicial, alfa y beta. A los tres le vamos a asignar a su atributo estado, el estado actual que se identifica como state, así como a alfa y beta le asignamos a sus heuríticas un valor muy negativo y otro muy positivo respectivamente antes de llamar a la función alfa-beta.

Dentro de la función principal alfa-beta primero comprobamos que si el estado del nodo a estudiar es uno final o si es nos encontramos a profundidad doce, ya que desde el inicio no podemos abrir todos los posibles estados que se pueden generar en este juego. Si no se cumple esta condición veremos si entonces comenzaremos creando un nodo auxiliar al cual se le añadirán la heuristica de alfa o beta y los movimientos siguientes posibles generados con la función Simulate-Move. Llamaremos a la recursividad para generar el nodo nuevo con la heurística correspondiente para posteriormente compararla con el nodo auxiliar, para lo cual utilizaremos la funcion isMax que nos devolverá un booleano diciendo si el primer valor pasado por parámetro es mayor que el que hemos pasado como segundo. Al estar en un comienzo en el J1 queremos que el valor de alfa sea el mayor valor posible entre el nodo auxiliar, el nuevo que proviene del proceso recursivo y el alfa que viene por parámetro, si entran en estas comprobaciones asignaremos las heuristica mayor y el movimiento actual ya que será el movimiento por el cual podremos llegar al . Posteriormente si el valor de la heuristica del nodo beta es menor o igual que el de alfa se realiza poda. El proceso será similar cuando le toque a min solo que en la función de isMax queremos tomar el valor mas pequeño para min.

4 Heurística

La función heurística que se ha utilizado es la diferencia entre las semillas que tenemos en nuestro granero menos las que el contrario tiene en el suyo, ademas sumaremos las semillas que tenemos en nuestro semillero, con el suplemento de que si hemos llegado a un estado final en el que ganamos le sumamos una cantidad determinada que en este caso va ser 10. A parte si repetimos tiro entonces se le sumará un total de 6. Finalmente para priorizar nuestro valor la multiplicaremos por tres, y devolveremos nuestro valor menos el del contrario mas nuestras semillas de los semilleros por dos.

4.1 Ejemplos de ejecución

A continuación se mostrarán ejemplos de ejecución del nuestro bot contra los dos bots que tenemos a nuestra disposición por parte de los profesores.

4.1.1 boTijo (P1) vs RandomBot (P2)

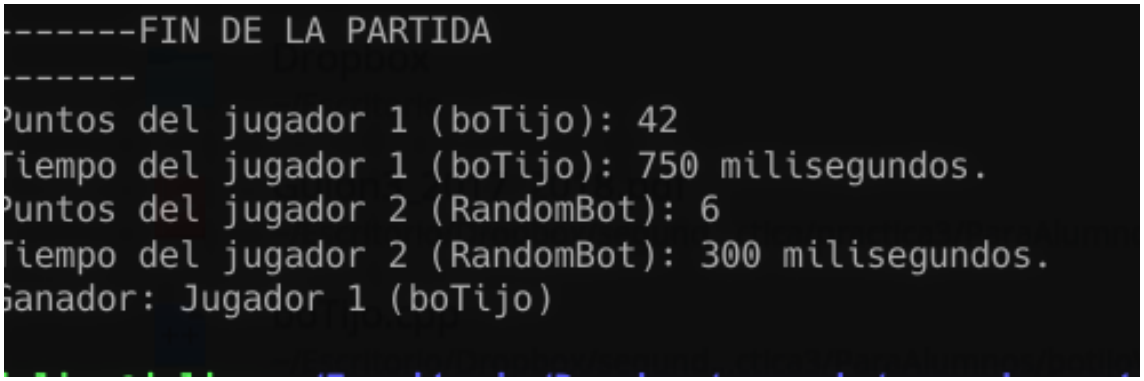
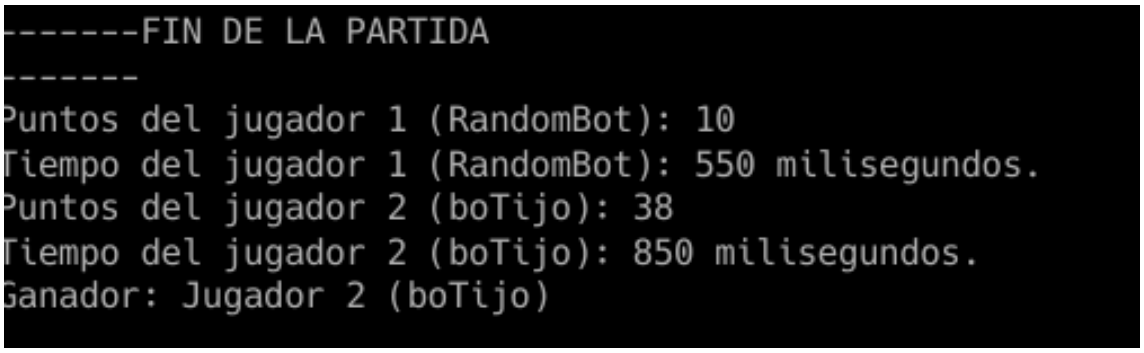
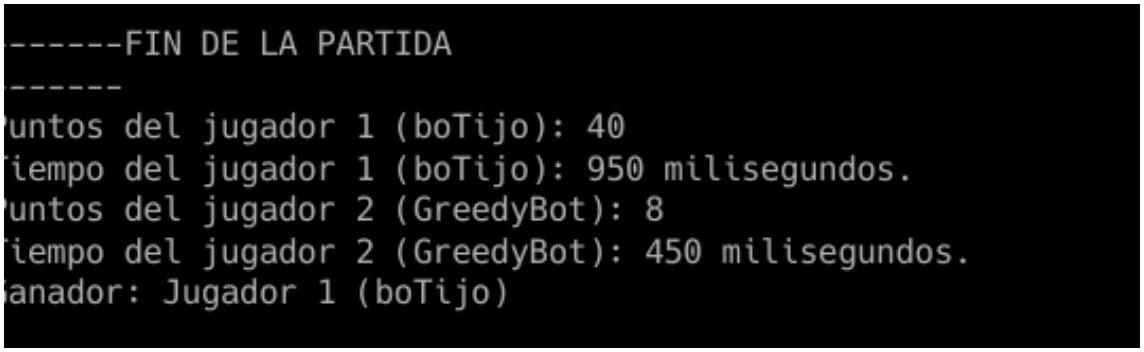


Figure 1: Resultado como jugador 1 contra RandomBot de jugador 2

4.1.2 boTijo (P2) vs RandomBot (P1)

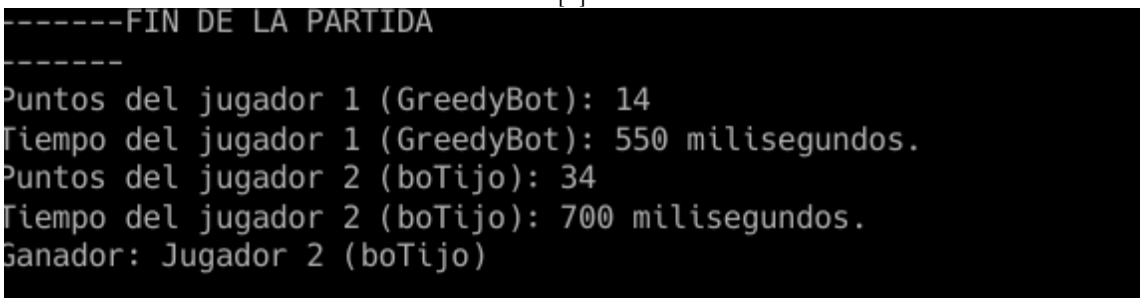


4.1.3 boTijo (P1) vs GreedyBot (P2)



4.1.4 boTijo (P2) vs GreedyBot (P1)

[h]



4.2 Conclusión

Esta práctica ha sido bastante instructiva ya que la necesidad de la recursividad en el algoritmo de poda alfa-beta me ha ayudado a asentar conocimientos sobre este tema. Además es interesante ver como este tipo de algoritmo podemos extrapolarlo a otros juegos por turnos.

He tenido problemas al haber cambiado el diseño normal del algoritmo por la utilización de nodos y la devolución de estos pero finalmente se ha conseguido realizar el algoritmo correctamente.

Si hubiese algún problema con el tiempo de ejecución del movimiento, ya que en mi ordenador no genera timeout y el servidor de mancala no está operativo y no se puede probar correctamente, simplemente bajando la profundidad el tiempo por movimiento se reduciría el tiempo.