INTELIGENCIA ARTIFICIAL
GRADO EN INGENIERÍA INFORMÁTICA
UNIVERSIDAD DE GRANADA

PRÁCTICA 3
Agentes en entornos con adversario

Sofía Fernández Moreno
23 de mayo de 2018

1. Diseño de "estado" para realizar la búsqueda

La búsqueda la realizaremos a través de nodos, mediante la clase GameState.
Usaremos árboles/grafos para representar posibles estados de un problema y los movimientos/acciones a realizar para pasar de un estado a otro.
Cada nodo del grafo representa un estado del problema, el cual representa las semillas de cada semillero.
Contiene toda la información necesaria para poder diferenciar el estado del problema de cualquier otro.
Las aristas se asocian a acciones o decisiones a tomar, los movimientos a realizar con las semillas.
Una acción (arista), aplicada sobre un estado del problema da como resultado un cambio de estado a otro conocido.

2. Algoritmo implementado

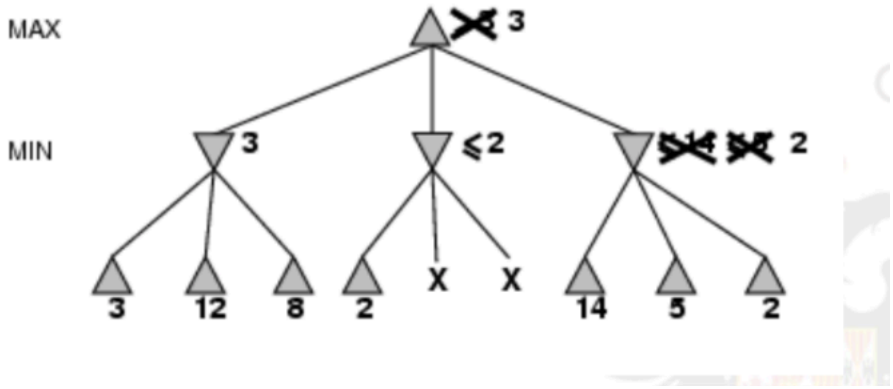
Para esta práctica he utilizado como algoritmo la técnica *Poda Alpha-Beta*, que según la teoría vista en clase es un algoritmo aplicado sobre el algoritmo *MiniMax*, por el cual nos permite con el mismo esfuerzo explorar un árbol de mayor profundidad.
En este algoritmo Alpha y Beta son inicializados a $-\infty$ y ∞ respectivamente, mientras que el jugador de turno del nodo es el jugador actual del algoritmo.
Ahora si el valor obtenido de la evaluación es más alto que alpha, actualizaremos alpha a ese valor.
Cuando el jugador de turno del nodo es el oponente, comprobamos que, si el mejor valor obtenido de la evaluación es menor que Beta, entonces Beta se actualiza a ese valor.
Hasta aquí sería igual que el algoritmo MiniMax.
Ahora para podar e identificarnos con este algoritmo, vamos comprobando que Beta es menor o igual que Alpha, en ese caso, el ciclo se detiene y las siguientes ramas se omiten, ya que los movimientos no son de nuestro interés.
Con todo lo anterior se espera mejorar el tiempo de cálculo, consumo de memoria y ser igualmente de robusto al Algoritmo MiniMax.

Para identificar cuando nos encontramos en un nodo MAX o en un nodo MIN, utilizaremos el estado de nuestro Jugador, es decir, su turno.

Cuando sea el jugador principal nos encontraremos en un nodo MAX. Para el oponente nos encontraremos en un nodo MIN.

Dentro de un nodo MAX, quiero calcular todos los posibles movimientos que se puedan hacer. Entonces para cada hijo del nodo state, se recorre todos los posibles movimientos, teniendo en cuenta que la profundidad pasada será profundidad-1 al bajar un nivel y que el jugador quiere maximizar y lo que se busca es minimizar.

Dentro de un nodo MIN, quiero calcular todos los posibles movimientos que se puedan hacer. Entonces para cada hijo del nodo state, se recorre todos los posibles movimientos, teniendo en cuenta que la profundidad pasada será profundidad-1 al bajar un nivel y que el jugador quiere minimizar y lo que se busca es maximizar.



3. Heurística utilizada

Aquí plantearemos la heurística para evaluar nuestro tablero, es decir, para cuando lleguemos a un estado final y cuando nuestra profundidad sea cero, es decir, nos encontremos en el último nodo.
Para esto tendremos que tener en cuenta que a la hora de ejecutar nuestro algoritmo la profundidad a pasar sea par o impar, ya que creará mayor preferencia a un jugador u otro.

Para mi heurística, intento obtener el mejor beneficio actualizado con cada movimiento anterior obtenido mediante la puntuación siguiente de cada movimiento, esto solamente en el caso de que nuestro numero de movimiento coincida con lo que guardamos en cada semillero, es decir, en un semillero concreto dependiendo la posición.
Para ello ese mejor beneficio se actualizará cuando recorramos todas las posiciones de nuestro tablero. Obteniendo aquel que tenga mejor beneficio que los movimientos ya obtenidos.
A la hora de devolver este valor, siempre comprobaremos que tengamos al menos un mejor valor actualizado, de no ser así devolveremos por defecto la puntuación del granero de nuestro jugador principal.

4. Pruebas Realizadas Contra GreedyBot

Valores obtenidos cuando J1=BotChui274

Valor Alfa-Beta con Profundidad = 16
16
17
23
24
24
37
37
37
37
38
39
39
39

5. Bibliografía

- BECERRA CORREA, Nelson. Aplicación de estrategias de búsqueda en el juego de Mancala, Tesis de maestría, Facultad de Ingeniería, Universidad Nacional de Colombia. 1998