

Diseño de Algoritmos – Algoritmos II

Nazareno Aguirre
Departamento de Computación
Facultad de Ciencias Exactas, Físico-Químicas y Naturales
Universidad Nacional de Río Cuarto

Clase 11: Búsqueda Adversaria

Búsqueda en Problemas con Adversarios

- ⌚ Las estrategias generales de búsqueda que hemos visto permiten resolver problemas en los que en la búsqueda es realizada por sólo un “jugador”.
- ⌚ En problemas en los cuales se compite contra un adversario en la búsqueda de estados exitosos, las técnicas de búsqueda vistas no son las apropiadas.

Características de los Problemas de Búsqueda con Adversarios

Podemos reconocer algunas características propias de los problemas de búsqueda con adversarios:

- Existe un estado inicial bien definido
- A partir del estado inicial, se avanza mediante reglas de avance bien definidas, alternando movimientos con un adversario
- El objetivo del adversario, al igual que el nuestro, es conseguir llegar a un estado exitoso
- En general si el adversario tiene éxito, nosotros fracasamos

A diferencia de los problemas de búsqueda convencionales, en problemas con adversarios el avance no depende de un solo “agente”.

Ejemplos de Problemas de Búsqueda con Adversarios

Los ejemplos clásicos de búsqueda con adversarios son juegos:

- ajedrez
- Ta-Te-Ti (tic-tac-toe)
- Damas
- Otello
- Go

La Técnica Min-Max

La técnica básica para solución a problemas de búsqueda con adversarios es Min-Max. Esta técnica se basa en considerar que el adversario puede realizar, cuando le toque, el mejor movimiento posible, y por lo tanto se independiza de las decisiones del adversario.

Al igual que para problemas de búsqueda convencionales, esta técnica se basa en la construcción de un árbol de juego.

Árboles de Juego

Los árboles de juego para búsqueda en problemas con adversarios se construyen de la siguiente forma:

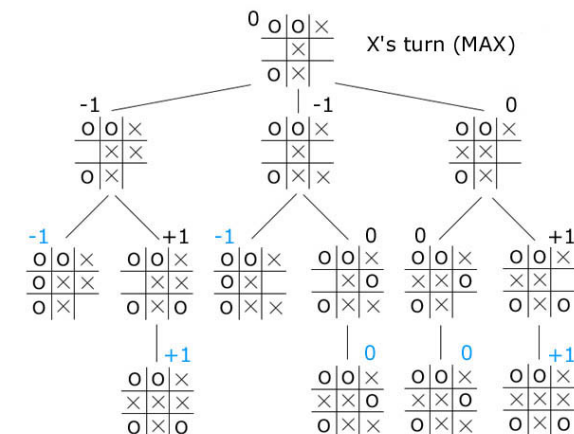
- Las configuraciones del juego son los nodos del árbol
- La raíz es el estado inicial del juego
- Si e' resulta de la aplicación de una de las reglas de avance del juego a partir de e , entonces e' es hijo de e en el árbol
- Los niveles del árbol se clasifican como Min o Max, según corresponda el movimiento al adversario o al jugador principal
- las hojas se etiquetan con +1, 0 o -1, según corresponda a una configuración en la que ganó el jugador principal, fue empate, o ganó el adversario
- Cada nodo interior del árbol en un nivel Max (resp. Min) se etiqueta con el máximo (resp. mínimo) de los valores de los hijos

Algoritmo Min Max

```

Funcion minMax(n) --> Valor
  Si n es hoja
    retornar valor(n)
  Sino
    x:= MIN_VAL
    y:= MAX_VAL
    Para cada hijo n_k de n hacer
      Si n es Max
        x:= max(x, minMax(n_k))
      Sino
        y:= min(y, minMax(n_k))
    Fsi
  Fpara
    Si n es Max
      retornar x
    Sino
      retornar y
  Fsi
Ffuncion
    
```

Un Ejemplo



Cómo usar el Árbol Min Max para Buscar una Estrategia Ganadora

Suponiendo que contamos con el árbol de juego, la mejor estrategia que podemos seguir es elegir la rama con mayor valor en los turnos en los que nos toque mover.

Por supuesto, esto lo podemos hacer si contamos con el árbol, el cual además debe ser finito (pues necesitamos los valores de los estados finales, la hojas, para etiquetar los nodos internos).

Adaptando Min-Max para uso en la Práctica

En general, los árboles de juego son de una gran dimensión, por lo cual, aún siendo finitos, no se los puede construir por completo.

Una adaptación útil para poder aplicar la técnica en la práctica se basa en la utilización de una función de valoración de configuraciones, y un límite k en el número de futuros movimientos a considerarse:

- En el momento en el que nos toque "jugar", generamos el sub-árbol de juego que tenga como raíz la configuración actual, y que tenga hasta k niveles
- valoramos las "hojas" de este sub-árbol usando la función de valoración (en el caso en que las hojas no sean ya estados finales)
- valoramos los nodos internos usando el procedimiento Min-Max

Poda Alfa-Beta: Una Mejora a Min-Max

Incluso utilizando las técnica Min-Max "adaptada", para límites k de movimientos futuros razonables el tamaño del sub-árbol de juego a explorar es en general muy grande.

Debido a esto, es necesario buscar mejoras al algoritmo Min-Max. Una optimización interesante es la provista por la técnica de poda Alfa-Beta.

Algoritmo Min-Max con Poda Alfa-Beta

```
Funcion minMaxAB(n, alfa, beta) --> Valor
  Si n es hoja o esta a nivel maximo
    retornar valor(n)
  Sino
    Para cada hijo  $n_k$  de  $n$  y mientras  $\alpha < \beta$  hacer
      Si  $n$  es Max
         $\alpha := \max(\alpha, \text{minMaxAB}(n_k, \alpha, \beta))$ 
      Sino
         $\beta := \min(\beta, \text{minMaxAB}(n_k, \alpha, \beta))$ 
      Fsi
    Fpara
    Si  $n$  es Max
      retornar  $\alpha$ 
    Sino
      retornar  $\beta$ 
    Fsi
  Ffuncion
```

Un Ejemplo

