

PODA ALFA-BETA

- Mejora del Algoritmo **Minimax**; aplicado en juegos de adversarios por turnos
- Se aplica en espacios de estados demasiado grandes como para analizar todos los nodos
- La información es imperfecta; es decir, no se conoce el estado del contrincante. P. ejem. En juegos donde no se ve el tablero del adversario

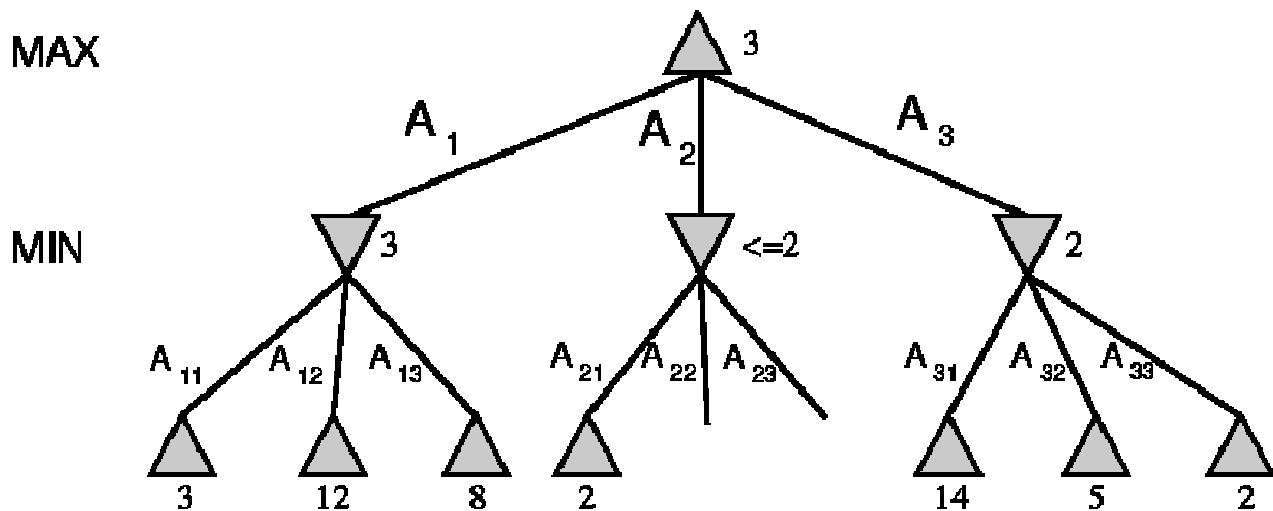
Características de la Poda

Alfa-Beta

- Omitir la expansión de nodos que por sus valores no pueden ser los mejores (peores).
- Interrumpe la búsqueda en algún nivel y aplica evaluaciones heurísticas a las hojas (profundidad limitada)
- Si el valor del nodo **MAX** (alfa) es menor que el más alto hasta este momento, entonces omitir nodo.
- SI el valor del nodo **MIN** (beta) es mayor que el nodo más bajo hasta el momento, entonces omitir nodo.
- **Alfa-Beta** permite búsqueda dos veces más profunda.

- Ordenamiento de los operadores, resultante del conocimiento o experiencia.
- Únicamente importa el orden y no los valores exactos.
- La poda no afecta al resultado final.
- **Alfa-Beta** es una mejora del algoritmo **Minimax** que evita revisar porciones dominadas del árbol, que no pueden proveer información útil sobre la jugada siguiente.
- **Alfa-Beta** es un algoritmo de búsqueda en profundidad, rama y cota, que avanza por el árbol en un orden ya fijado (p.ej., de izquierda a derecha) y va usando la información de la valuación de los nodos hoja para podar ramas dominadas que no sirven

para cambiar el valor **Minimax** del nodo inicio (la jugada inminente).



Estructuras de datos

Dos variables deben recordarse a lo largo de la búsqueda:

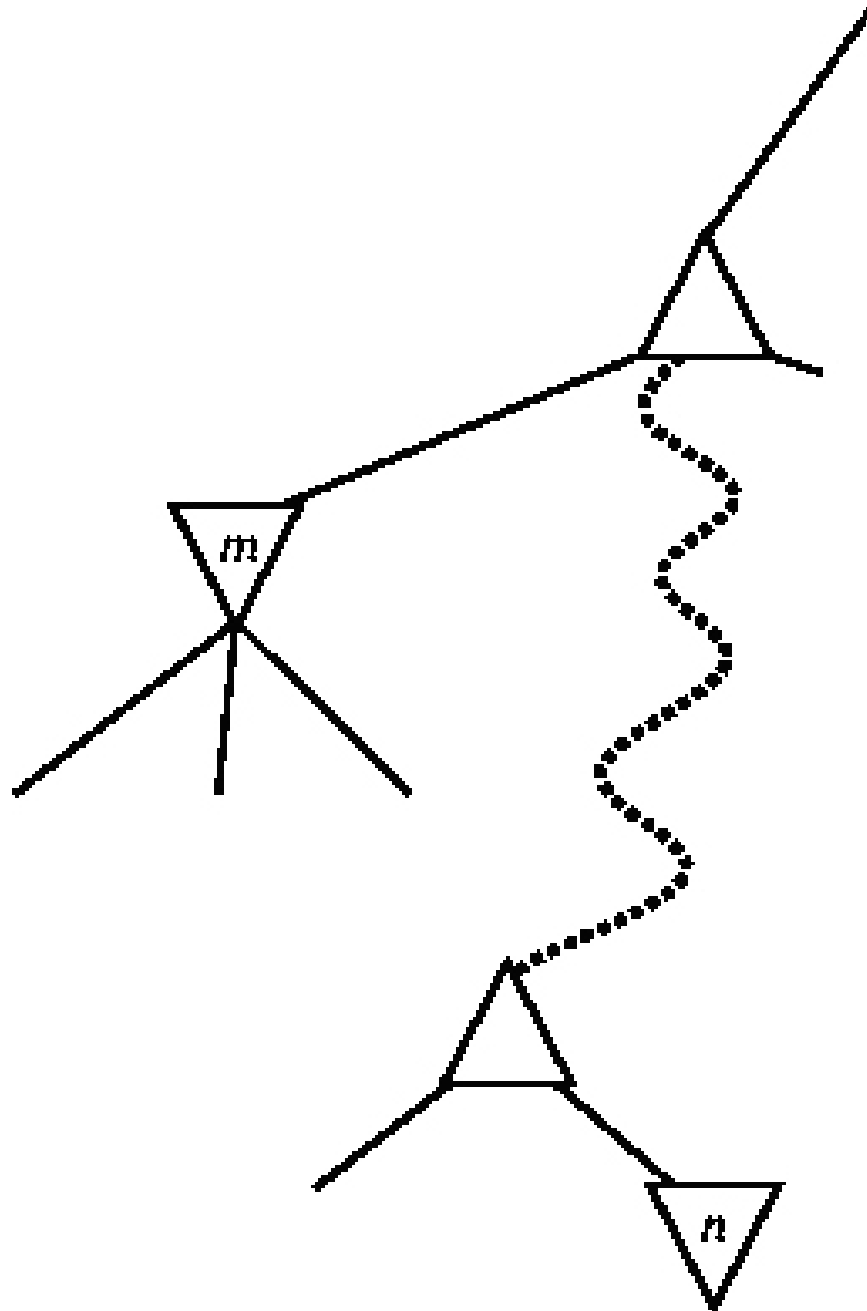
- **Alfa:** Límite inferior encontrado hasta ese momento.
- **Beta:** Límite superior.

- En los niveles maximizantes donde **MAX** debe optar, sólo **Beta** se usa para podar la búsqueda
- En los niveles minimizantes donde **MIN** debe optar, sólo **Alfa** se usa para podar.
- **Alfa-Beta** es el algoritmo más usado para buscar en árboles de juegos.

Origen del nombre Alfa

Alfa es el nombre del mejor valor **m**, para **MAX**, encontrado hasta ahora en su ruta de búsqueda en un nivel de **MIN**.

- Si **n** es peor que **Alfa**, **MAX** lo evitará
⇒ podar esa rama punteada
- **m** y **n** son nodos de **MIN**



Algoritmo de búsqueda Alfa-Beta

Corresponde a la combinación de tres aportes:

- Ejecutar Minimax
- Mantener recordados **Alfa** y **Beta**
- Podar

function MAX-VALUE(*state*, *game*, α , β) returns the minimax value of *state*

inputs: *state*, current state in game

game, game description

α , the best score for MAX along the path to *state*

β , the best score for MIN along the path to *state*

if CUTOFF-TEST(*state*) then return EVAL(*state*)

for each *s* in SUCCESSORS(*state*) do

$\alpha \leftarrow \text{MAX}(\alpha, \text{MIN-VALUE}(s, \text{game}, \alpha, \beta))$

if $\alpha \geq \beta$ then return β

end

return α

function MIN-VALUE(*state*, *game*, α , β) returns the minimax value of *state*

if CUTOFF-TEST(*state*) then return EVAL(*state*)

for each *s* in SUCCESSORS(*state*) do

$\beta \leftarrow \text{MIN}(\beta, \text{MAX-VALUE}(s, \text{game}, \alpha, \beta))$

if $\beta \leq \alpha$ then return α

end

return β

Comportamiento de los nuevos algoritmos de búsqueda en juegos

- **Alfa-Beta** busca solamente $\frac{3}{4}b$ de los b movimientos posibles desde una posición dada de juego.
- Esto significa que la profundidad de búsqueda se puede incrementar por un factor $= \log b / \log \frac{3}{4}b \approx 4/3$ por encima de una búsqueda exhaustiva **Minimax**.
- B es aquí el factor de ramificación efectivo.
- Si los sucesores se ordenan a la perfección (definido como que al usar **Alfa-Beta** la búsqueda es mínima),

Alfa-Beta examina $2bd/2 - 1$
posiciones de juego.

- Así tenemos

$B = b$	en búsquedas exhaustivas
$B \sim 3/4b$	en Alfa-Beta ordenando al azar
$B = 1/2b$	en Alfa-Beta ordenando perfectamente