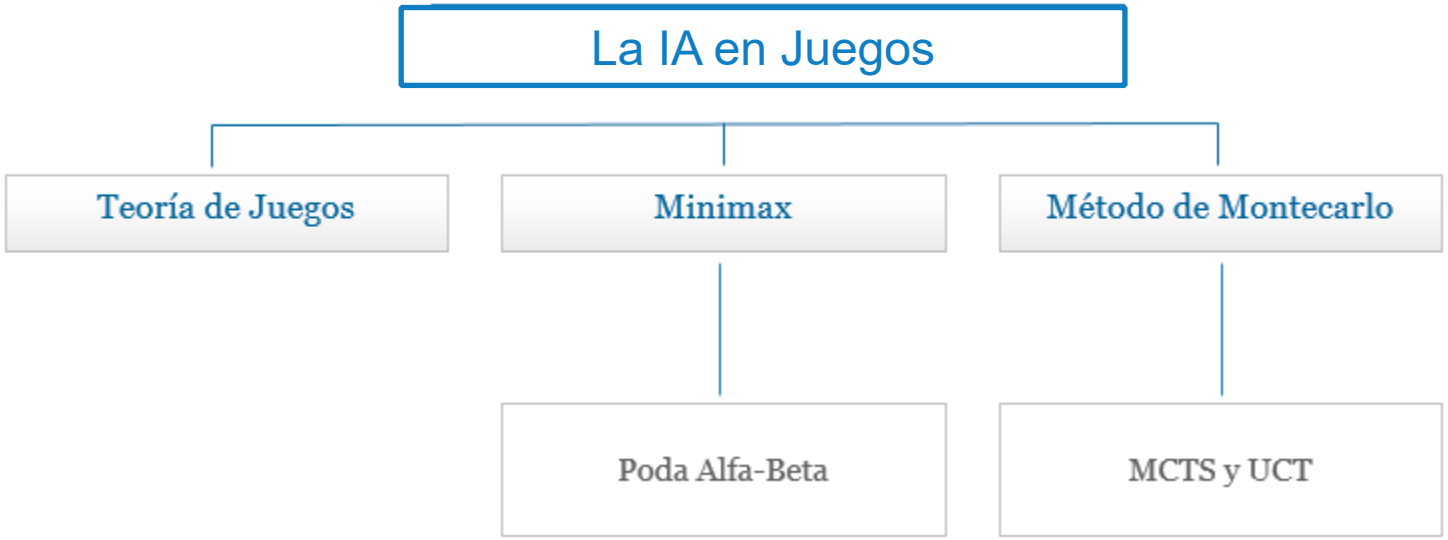
Elena Verdú Pérez

La Inteligencia Artificial en Juegos

¿Cómo estudiar este tema?



» TEMA 5. LA INTELIGENCIA ARTIFICIAL EN JUEGOS [Esquema Tema]

IDEAS CLAVE	LO + RECOMENDADO	+ INFORMACIÓN	TEST
¿Cómo estudiar este tema? Teoría de Juegos Minimax y poda Alfa-Beta Simulación: Monte Carlo	**Lecciones magistrales** TV La Inteligencia Artificial en Juegos **No dejes de ver...** TV Minimax en OpenCourseware TV Ejemplo de Poda Alfa-Beta por Subbarao Kambhapati	**A fondo** Teoría de Juegos en Coursera AI Game Developer Minimax y poda Alfa-Beta en Wikipedia **Bibliografía** **Recursos externos** Stockfish: jugador automático de ajedrez	

Objetivos de la clase

- Enumerar características que definen los juegos.
- Representar juegos no cooperativos en forma normal y en forma extensiva.
- Aplicar el algoritmo minimax.
- Aplicar el algoritmo minimax con poda alfa-beta.
- Describir el método de simulación Monte-Carlo.

Teoría de juegos

Estudio de los métodos matemáticos de conflicto y cooperación entre agentes racionales deliberativos

Los juegos estudiados por la teoría de juegos **se definen matemáticamente en función de:**

- ❑ Número de jugadores
- ❑ Información disponible
- ❑ Acciones que cada jugador puede realizar
- ❑ Recompensa obtenida

Teoría de juegos

Otras características relevantes:

- Secuencial/simultáneo →los jugadores juegan por turnos o todos a la vez
- Suma constante → la recompensa que recibe un jugador se resta de la de otros
- Cooperativo / no cooperativo → los jugadores comparten un mismo objetivo y deciden en consenso o no
- Simétrico / asimétrico → los jugadores pueden realizar idénticas acciones y recibir idénticas recompensas o no

Teoría de juegos

Otras características relevantes:

- Duración finita/infinita
- Discreto / Continuo → número de estrategias y de estados finito o no
- Determinista / Estocástico →el resultado de una acción y su recompensa es determinista o fijo

Juegos clásicos adversariales: dos jugadores, información completa, recompensa infinita por victoria, secuencial, suma cero, no cooperativo, simétrico, duración finita, discreto y determinista

Teoría de juegos

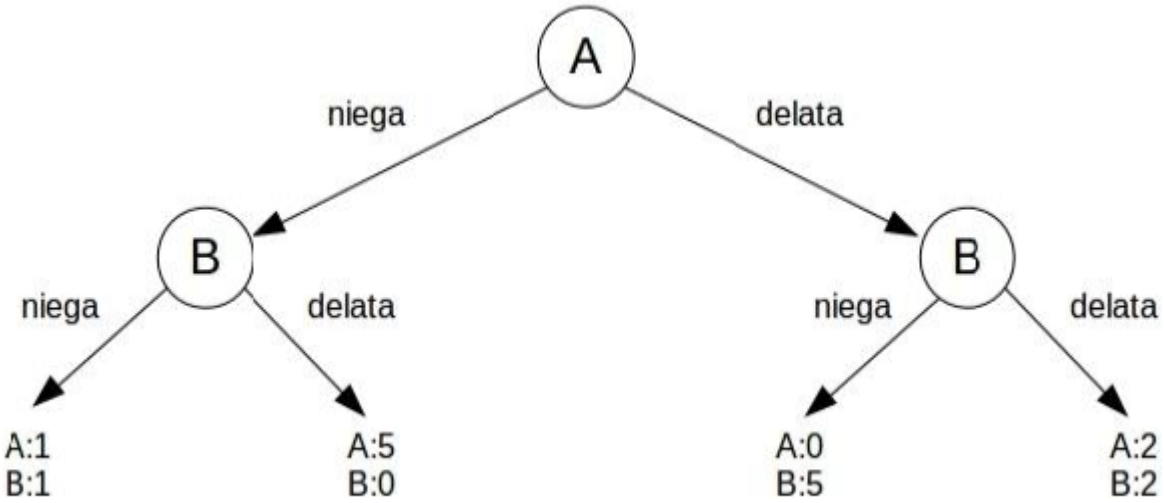
Los juegos no cooperativos se suelen describir de dos maneras: forma normal o forma extensiva

- La forma normal representa con una matriz *n*-dimensional el producto de las estrategias de los jugadores y la recompensa que cada jugador recibe.
- Juego del dilema del prisionero.

	Prisionero B niega	**Prisionero B delata**
Prisionero A niega	A: 1 años B: 1 años	A: 5 años B: Libre
Prisionero A delata	A: Libre B: 5 años	A: 2 años B: 2 años

Teoría de juegos

- Los juegos en los que la recompensa se obtiene después de una secuencia de acciones se representan de forma extensiva.



Teoría de juegos

Equilibrio de Nash: situación en un juego de 2 o más jugadores que asume que:

- Cada jugador conoce y adopta su mejor estrategia
- Todos los jugadores conocen las estrategias de los otros

Los jugadores aplican estrategias que maximizan sus ganancias dadas las estrategias de los otros jugadores.

Cada jugador no modificará su estrategia mientras los otros mantengan las suyas.

Minimax (caso de un juego muy sencillo)

- Se consideran 2 jugadores: MIN y MAX.
- El objetivo es que gane MAX.
- MAX intentará maximizar su ventaja y minimizar la de MIN.
- MIN intentará hacer lo peor para MAX.

Minimax (caso de un juego muy sencillo)

- Construye un árbol que representa todas las posibles jugadas con nodos MIN o MAX.
  - *Normalmente los problemas no son sencillos y se limita la profundidad del árbol de búsqueda*

- Asignar a los nodos finales un valor
  - + ∞ → victoria MAX
  - - ∞ → victoria MIN

- Propagar valores hacia la raíz
  - Nodo MAX → valor máximo de hijos
  - Nodo MIN → valor mínimo de hijos

- Camino de victoria: uniendo los nodos de valor + ∞

Minimax (caso de un juego muy sencillo)

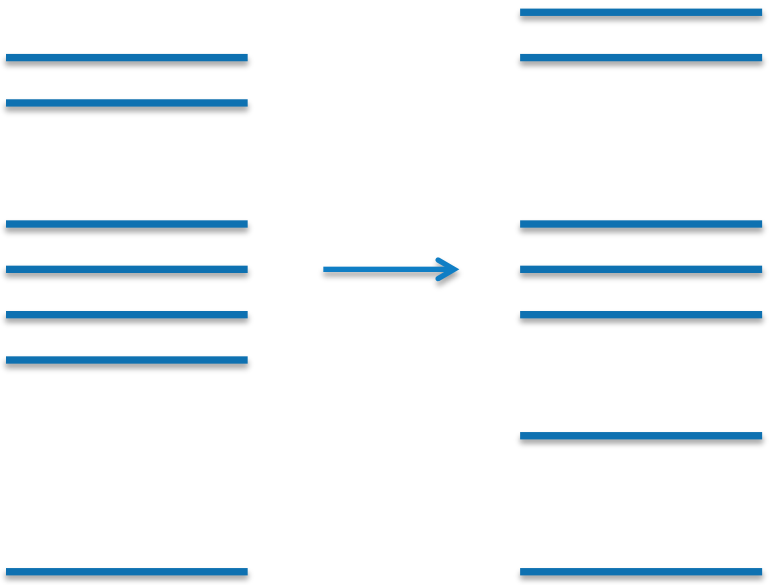
EJEMPLO: juego de los palillos

Jugador A

Jugador B

Minimax (caso de un juego muy sencillo)

EJEMPLO: juego de los palillos

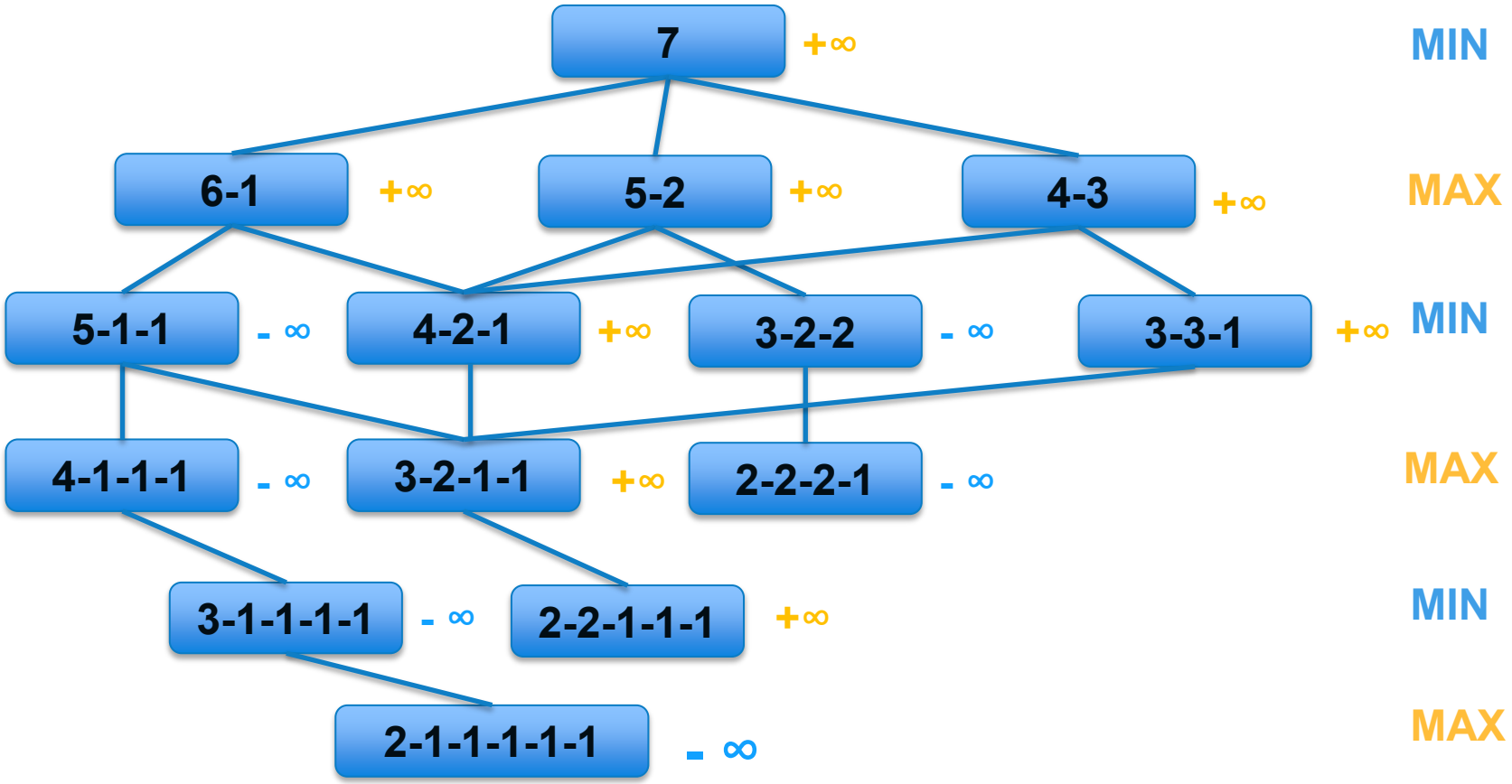


Jugador A

Jugador B

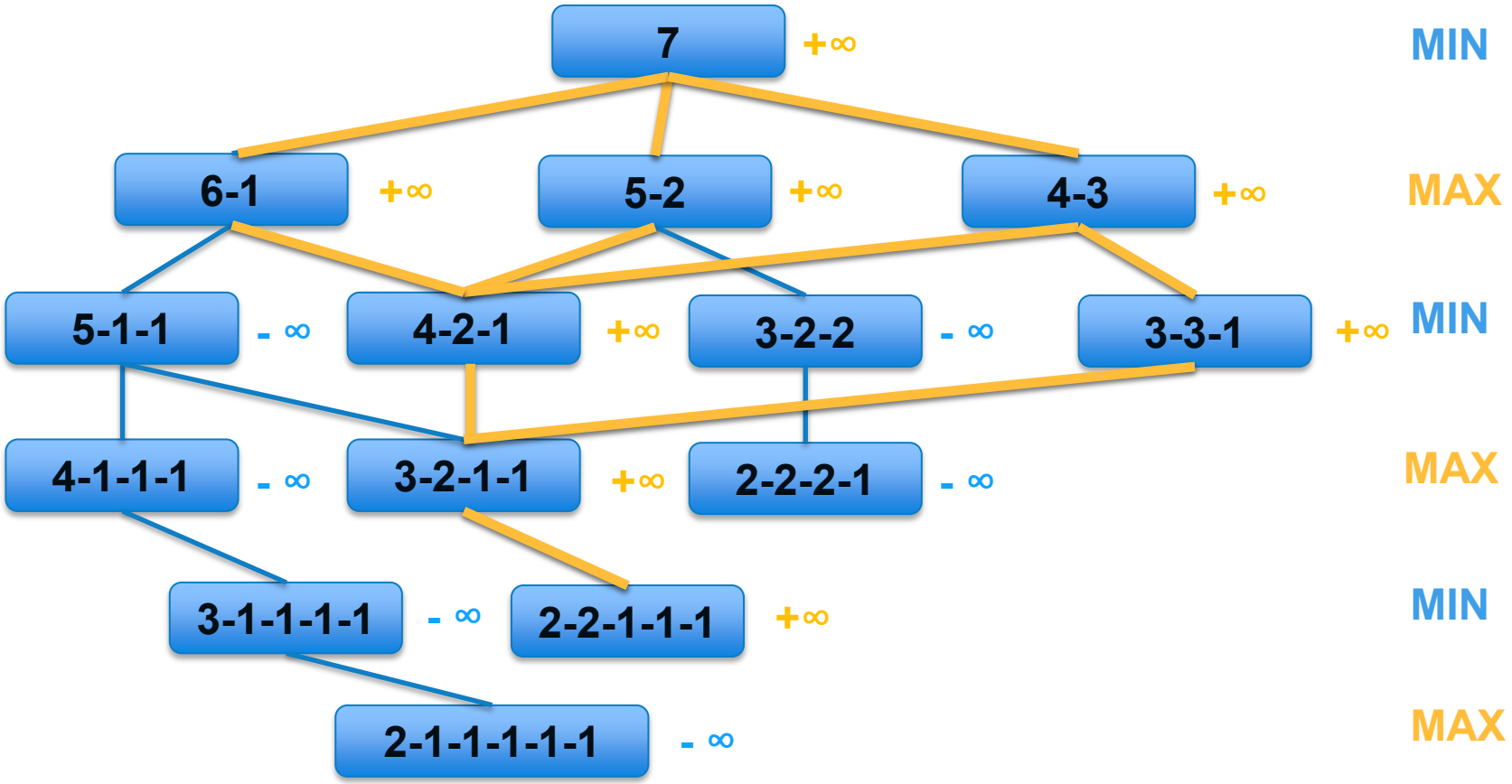
Minimax (caso de un juego muy sencillo)

EJEMPLO: juego de los palillos



Minimax (caso de un juego muy sencillo)

EJEMPLO: juego de los palillos



Minimax

Algoritmo de búsqueda en profundidad acotada diseñado para juegos:

- Secuenciales
- Deterministas
- Con información completa
- Discretos
- De suma cero
- De 2 jugadores

Minimax

Se asume que:

- ❑ No se explora el árbol al completo
- ❑ El rival siempre jugará la jugada óptima

- ❑ Los valores de recompensa van de +∞ a -∞
- ❑ Nodo hoja terminal toma el valor:+∞ o -∞
- ❑ Nodo hoja no terminal: Aplicar función de evaluación
- ❑ Estrategia jugador MAX es maximizar la recompensa
- ❑ Estrategia jugador MIN es minimizar la recompensa

Minimax

Dos tipos de nodos

- Nodos en los que juega MAX: se elige la opción que maximiza la recompensa (de MAX)
- Nodos en los que juega MIN (contrincante de MAX): se elige la opción que minimiza la recompensa (de MAX)

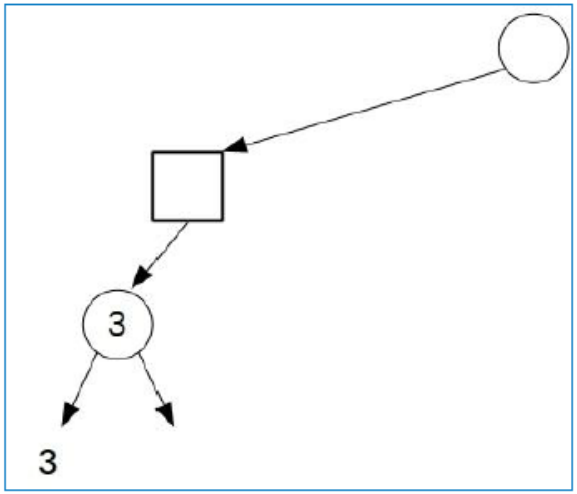
- El árbol de búsqueda de Minimax se explora en profundidad hasta alcanzar el límite de jugadas o dar con un nodo terminal.
- El valor de la recompensa se propaga hacia arriba.

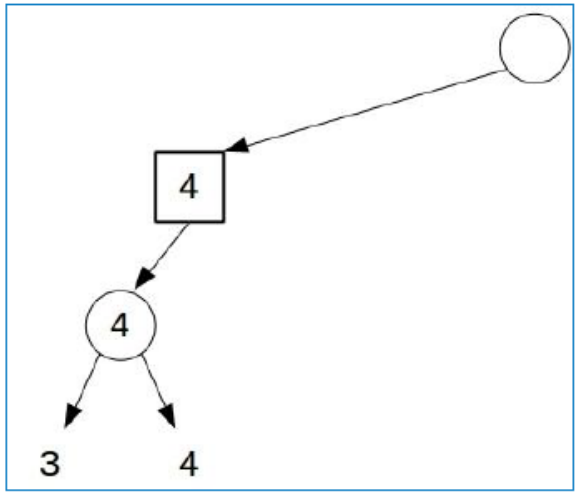
Minimax

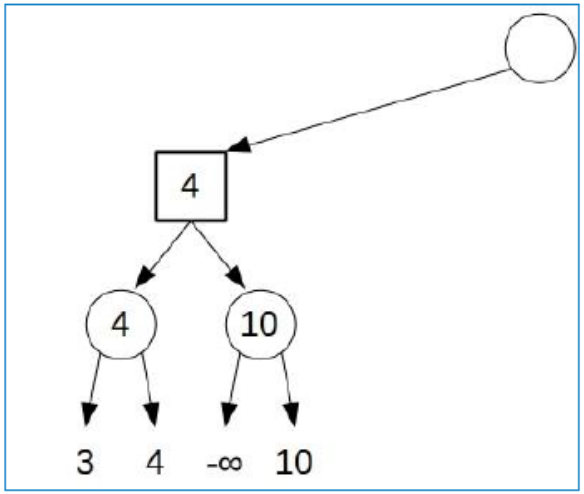
EJEMPLO: Tenemos un límite de 3 niveles (3 jugadas)

Jugador MAX – nodos redondos

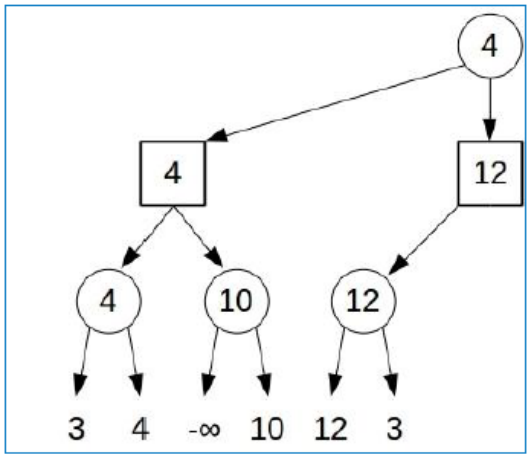
Jugador MIN – nodos cuadrados

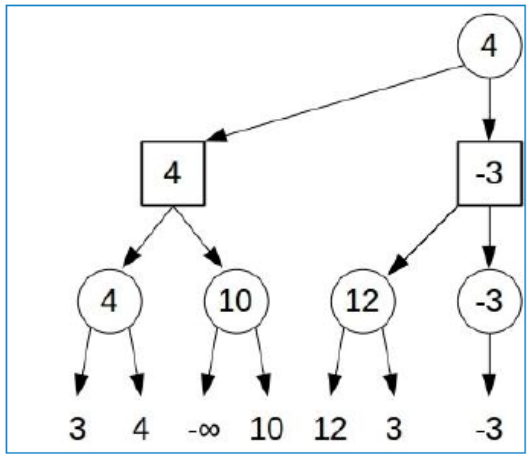


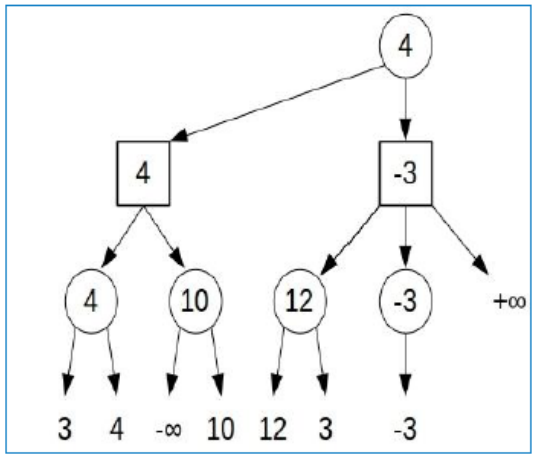


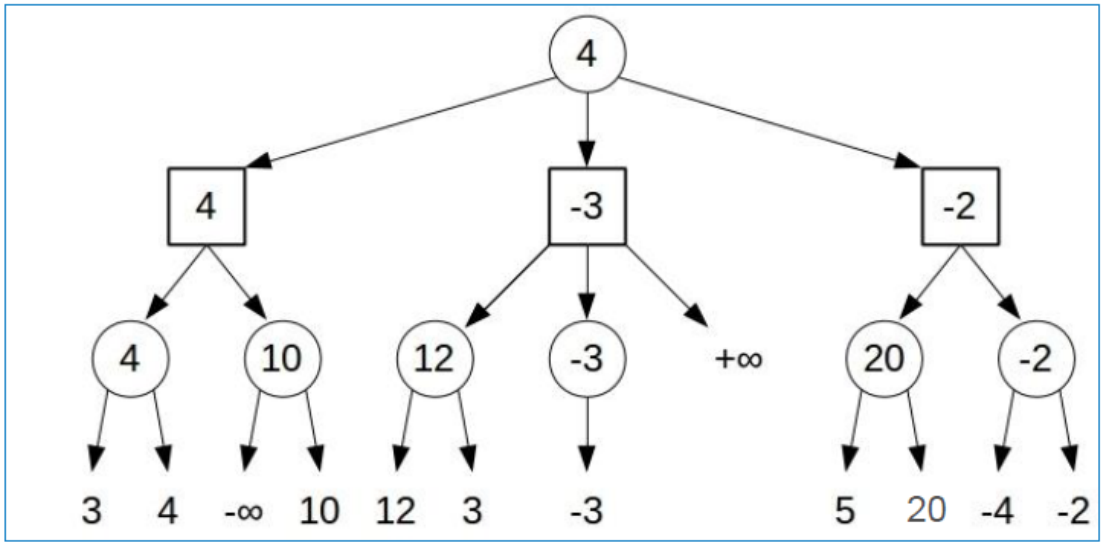


Minimax


4
4
12
4
10
12
3
4
-∞
10
12
3


4
4
-3
4
10
12
-3
3
4
-∞
10
12
3
-3


4
4
-3
4
10
12
-3
+∞
3
4
-∞
10
12
3
-3


4
4
-3
-2
4
10
12
-3
+∞
20
-2
3
4
-∞
10
12
3
-3
5
20
-4
-2

Minimax y poda Alfa-Beta

Técnica que permite dejar de buscar en un subárbol cuando se prueba que la recompensa que se va a obtener va a ser menor que la de otro subárbol ya explorado.

α – cota asociada con nodos MAX que nunca decrece

- se actualiza en nodos MAX
- máximo de las cotas mínimas

β – cota asociada con nodos MIN que nunca se incrementa

- se actualiza en nodos MIN
- mínimo de las cotas máximas

La recompensa actual de un nodo que sea relevante a la búsqueda tiene que estar entre α y β

Minimax y poda Alfa-Beta

- Desde el nodo raíz se llama recursivamente a los sucesores
  - Nodo terminal: devuelve el valor de función de evaluación
  - Nodo MAX: devuelve alfa y poda si alfa >= beta
  - Nodo MIN: devuelve beta y poda si alfa >=beta

- **Se poda por debajo de un nodo que tenga un valor de alfa mayor o igual que el valor de beta**

```
minimax-alfabeta (nodo N, profundidad, α, β, MAX)
1. Si profundidad = 0 O N es terminal
Entonces Devolver valor
2. Si MAX = Cierto
   Entonces Para cada sucesor S de N
      α ← max(α, minimax-alfabeta (S, profundidad-1, α,
β, Falso))
        Si β ≤ α
              Entonces Break
      Devolver α
3. Si No
   Entonces Para cada sucesor S de N
      β ← min(β, minimax-alfabeta (S, profundidad-1, β,
β, Cierto))
          Si β ≤ α
             Entonces Break
  Devolver β
```

Minimax y poda Alfa-Beta

Si es nodo MAX:
-se actualiza alfa
-Se devuelve el valor de alfa

Se llama con este mismo método a cada sucesor (y eso sucede recursivamente) y se les pasa alfa y beta .
Un nodo MAX al realizar esta llamada a un sucesor, su sucesor le devolverá el valor de su beta que se utilizará para actualizar el valor de alfa del nodo MAX (la beta quedará igual), siempre que supere el valor de alfa del

```
minimax-alfabeta (nodo N, profundidad, α, β, MAX)
1. Si profundidad = 0 O N es terminal
Entonces Devolver valor
2. Si MAX = Cierto
   Entonces Para cada sucesor S de N
     α ← max(α, minimax-alfabeta (S, profundidad-1, α, β, Falso))
       Si β ≤ α
              Entonces Break
     Devolver α
3. Si No
   Entonces Para cada sucesor S de N
      β ← min(β, minimax-alfabeta (S, profundidad-1, α, β, Cierto))
        Si β ≤ α
           Entonces Break
  Devolver β
```

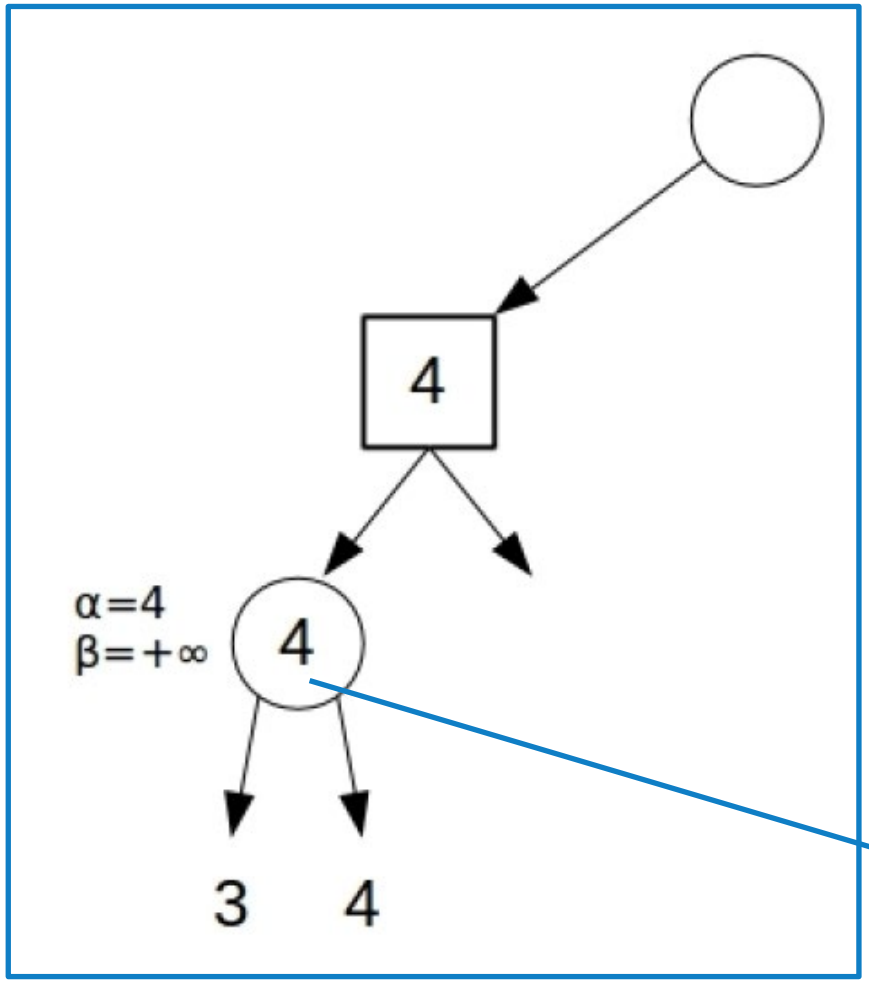
Poda si alfa >= beta

Si es nodo MIN:
-se actualiza beta
-Se devuelve el valor de beta

Beta se actualiza con el valor mínimo de los valores devueltos por sus sucesores (que son las alfas de los nodos MAX hijos)

Minimax y poda Alfa-Beta

- Inicialmente α=-∞ y β=+∞




```
minimax-alfabeta (nodo N, profundidad, α, β, MAX)
1. Si profundidad = 0 O N es terminal
Entonces Devolver valor
2. Si MAX = Cierto
   Entonces Para cada sucesor S de N
      α ← max(α, minimax-alfabeta (S, profundidad-1, α,
β, Falso))
         Si β ≤ α
                Entonces Break
      Devolver α
3. Si No
   Entonces Para cada sucesor S de N
      β ← min(β, minimax-alfabeta (S, profundidad-1, β,
β, Cierto))
          Si β ≤ α
             Entonces Break
  Devolver β
```

Nodo MAX→ Actualizamos Alfa
α = 4 (máximo de las cotas mínimas)

Minimax y poda Alfa-Beta

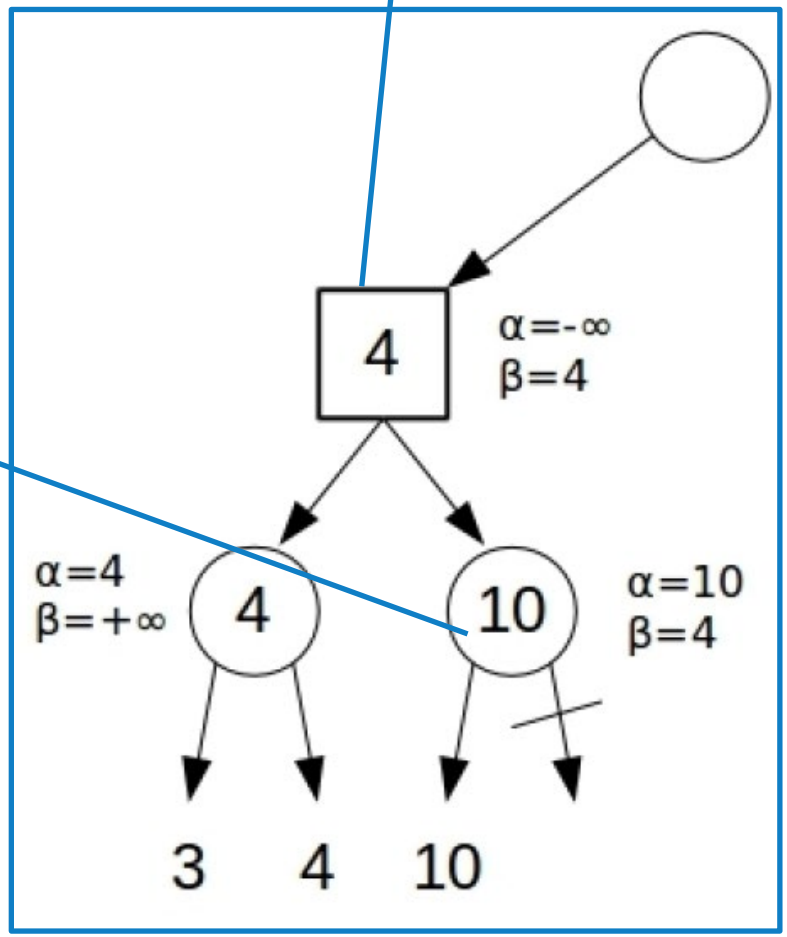
α y β se propagan desde nodos padres a sucesores

Nodo MAX: Actualiza α
α = 10 (máximo de las cotas mínimas)
β = 4 (valor propagado desde el padre)

Como α es mayor que β, se poda.

Ese nodo como mínimo vale 10 dado que es un nodo MAX, sabemos que el nodo padre MIN preferirá el nodo de valor 4 (β = 4) y por tanto no hace falta seguir explorando este nodo

Nodo MIN: Actualiza Beta
β=4 (mínimo de las cotas máximas)



Minimax y poda Alfa-Beta

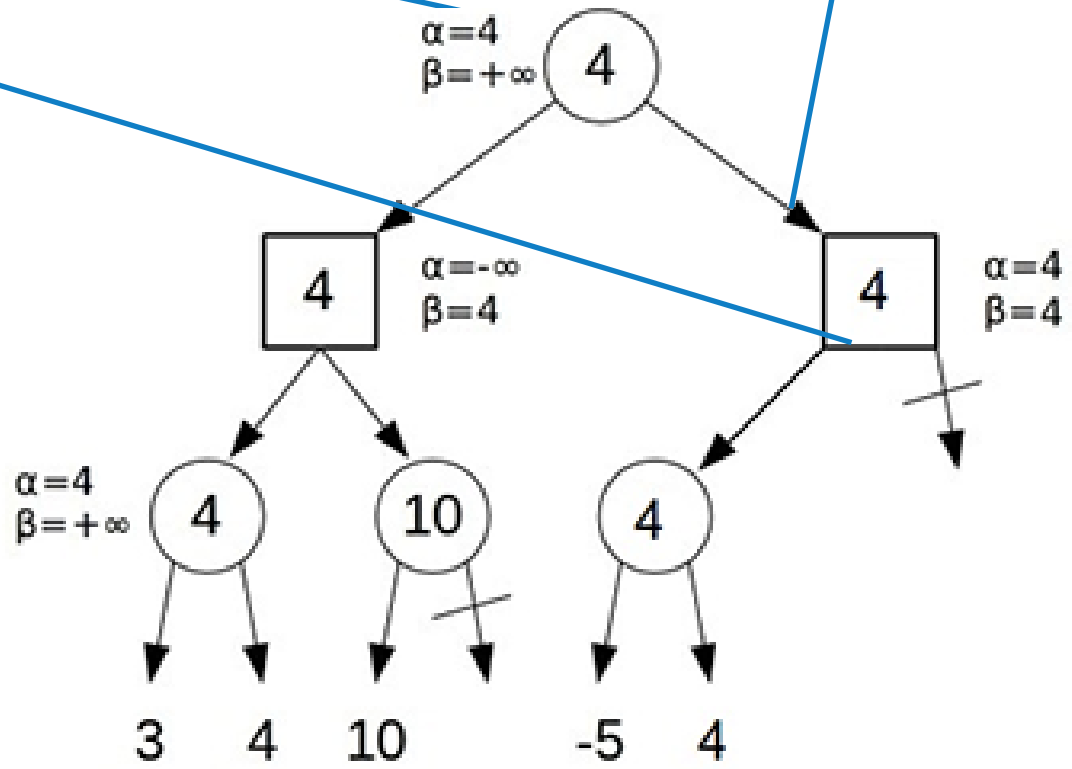
Se propaga el valor de α hacia sucesores

Nodo MAX: actualiza α

Nodo MIN que siempre tomará un valor menor o igual a β = 4

α = 4 nos indica que hay otro subárbol que será preferido por el nodo padre MAX a este, cuyo valor como mucho es 4.

Como α es mayor o igual que β, se poda



Se poda por debajo de un nodo que tenga un valor de alfa mayor o igual que el valor de beta.

Simulación Monte-Carlo. MCTS

Explorar el árbol de búsqueda de forma probabilística

Consiste en recrear múltiples simulaciones de partidas de forma total o parcial desde el punto de decisión, y, a partir de lo observado, elegir la rama más prometedora

Búsqueda de árbol Monte Carlo (MCTS, *Monte Carlo Tree Search*) :

- Construye un árbol de búsqueda dando prioridad a las ramas más prometedoras
- El sesgo que MCTS introduce está basado tanto en el número de veces en el que se ha hecho la simulación desde algún determinado subárbol como la proporción de simulaciones favorables al jugador

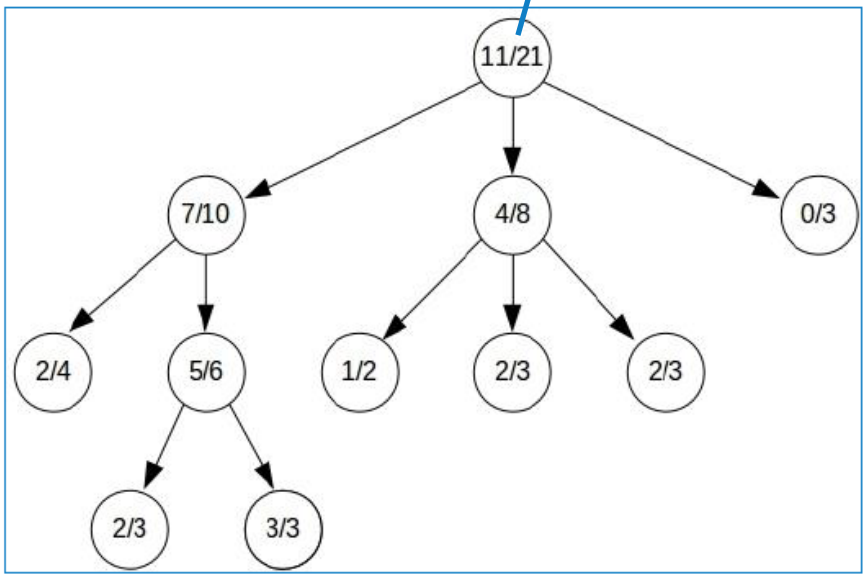
Simulación Monte-Carlo. MCTS

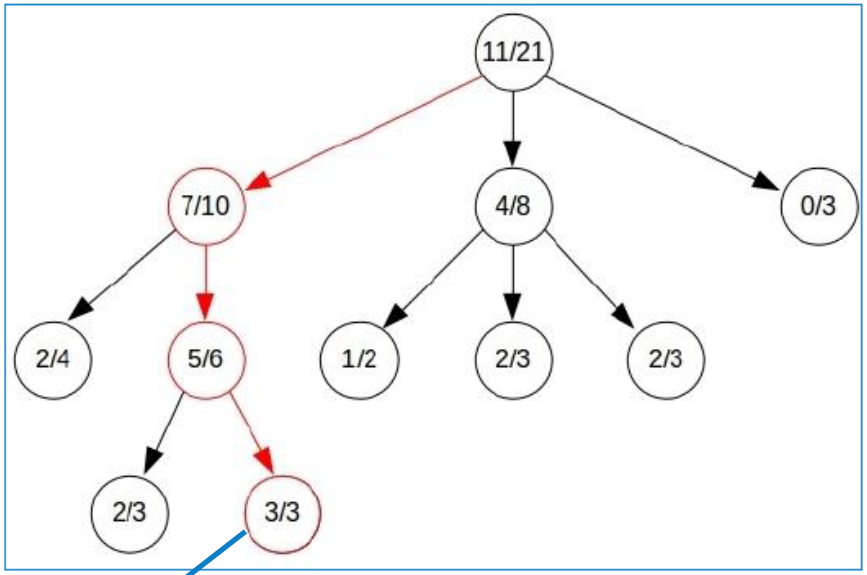
MCTS itera dando 4 pasos:

1. Baja por el árbol seleccionando en cada punto el nodo más prometedor.
2. Cuando llega a un nodo hoja, genera los sucesores y elige uno o más al azar.
3. Desde cada nuevo nodo, ejecuta una simulación.
4. Teniendo en cuenta el resultado de la simulación, actualiza el valor de los nodos que van de la raíz al nuevo nodo.

Simulación Monte-Carlo. MCTS

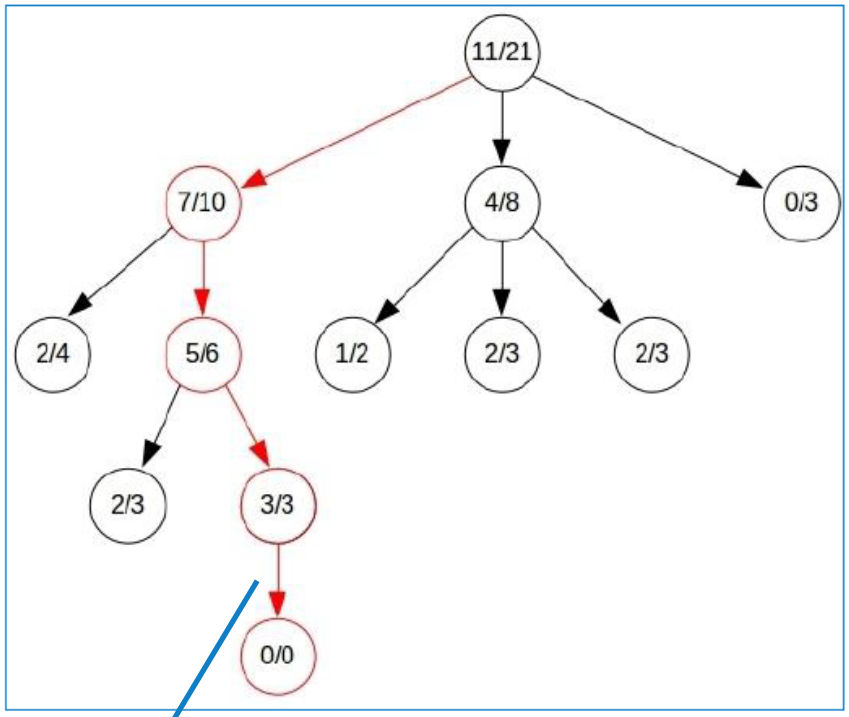
Cada nodo está etiquetado con:
veces en las que la simulación de alguno de los nodos de su subárbol ha sido favorable al primer jugador / total de simulaciones en el subarbol



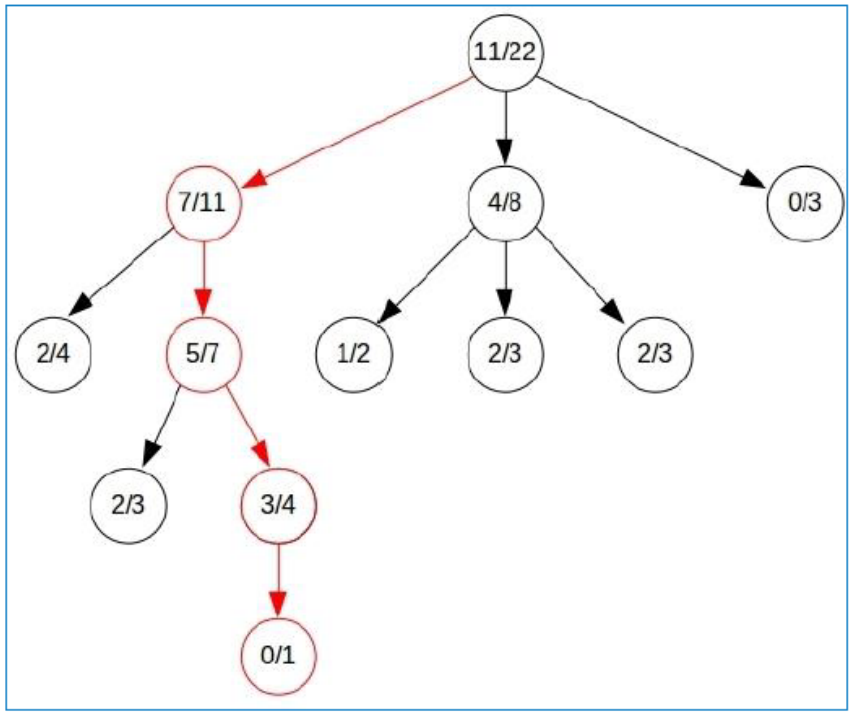


Se desciende por el camino más prometedor hasta llegar a un nodo hoja.
El camino elegido depende de las simulaciones favorables en función de las totales

Simulación Monte-Carlo. MCTS



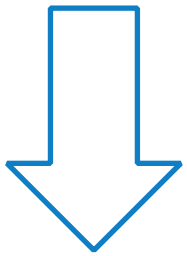
Se añade un hijo al azar y se ejecuta una simulación a partir de este estado



El resultado de la simulación se agrega en nodos ancestros

Simulación Montercarlo. MCTS

Esta forma de exploración del árbol puede provocar explorar un árbol muy en profundidad sin dar oportunidad de analizar otras ramas



La elección del nodo debe tener en cuenta el nº de simulaciones, independientemente del ratio, para lograr un balance entre la explotación de la rama más prometedora y la exploración del árbol en su conjunto visitando ramas poco visitadas.

Simulación Montercarlo. MCTS

UCT (Upper Confidence Bound 1 applied to trees)
→ determina la probabilidad de un nodo de ser elegido entre sus hermanos

$$\frac{w_i}{n_i} + c\sqrt{\frac{ln(t)}{n_i}}$$

- n_i → simulaciones ocurridas en el subárbol del nodo *i*.
- w_i → simulaciones favorables ocurridas en el subárbol del nodo *i*.
- t → simulaciones ocurridas en el subárbol del nodo padre.
- c → parámetro que balancea exploración y explotación ($\sqrt{2}$).

  Mayor c → mayor exploración – Menor c → mayor explotación.

Juego CompetencIA

- Desafío 8: Referente al tema 5, responde brevemente a las siguientes cuestiones ¿Qué características tienen los juegos en que se aplica Minimax? ¿Qué asume Minimax?

Los juegos en que se aplica Minimax son secuenciales, deterministas, con información completa, discretos, de suma cero y de dos jugadores enfrentados entre sí.
Minimax asume que no se puede explorar el árbol de búsqueda al completo, por lo que el algoritmo busca la solución teniendo en cuenta un número de jugadas limitado. También asume que el rival jugará la jugada óptima.

Juego CompetencIA

Desafío 10: Relativo al tema 6, contesta utilizando como mucho 100 palabras:¿qué diferencia fundamental existe entre los ejemplos con los que se construye un modelo de aprendizaje supervisado y los ejemplos con los que se construye uno de aprendizaje no supervisado? ¿qué objetivo se busca al construir un modelo de aprendizaje supervisado? ¿qué objetivo se busca al construir un modelo de aprendizaje no supervisado?

Formulario disponible en:
https://forms.office.com/r/yUwr2A3Sdf

¿Dudas?

¡Muchas gracias por vuestra atención!

¡Feliz y provechosa semana!

www.unir.net