Razonamiento y planificación automática

Alejandro Cervantes

Tema 6 (II): Poda alfa-beta y expectiminimax

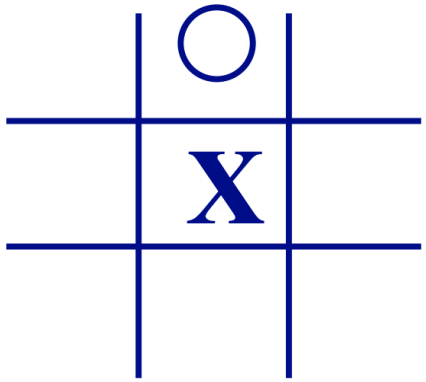
Índice

Minimax informado

Función de utilidad parcial

Se puede evaluar SIN LLEGAR A LA PROFUNDIDAD MÁXIMA analizando el estado (similar a la definición de heurística)

Función de evaluación f(n) parcial para "Tres en raya": número de posibilidades de hacer tres en raya más adelante del jugador X, menos número de posibilidades de hacer tres en raya del oponente O.



En esta posición X tiene 6 posibles líneas para hacer, y O tiene 4:

f(n) = 6 – 4 =2

Algoritmo minimax (informado)

1. Generamos el árbol de juego completo, hasta cierta profundidad

2. Aplicamos en cada nodo terminal (al final) la función de utilidad parcial

    2.1. Nodo max: se elige la acción (rama) que me lleva a utilidad máxima

    2.2. Nodo min: se elige la acción que lleva a utilidad mínima

3. Propagar las utilidades hacia arriba etiquetando los nodos con el valor de 2

4. Solución: en el nodo raíz (por convención es max) se escoge el mejor camino, y se sigue así hasta el resultado

Este recorrido se puede hacer de cualquier forma, ya que expandimos completamente el árbol; pero en la próxima clase lo haremos en profundidad para introducir la poda

Minimax con información parcial y profundidad máxima

Minimax

Valores en caso de **estado meta o empate**

$$f(n) = \begin{cases} +\infty & \text{si n es una situación ganadora} \\ -\infty & \text{si n es una situación perdedora} \\ 0 & \text{si n es una situación de empate} \\ f_{ev}(n) & \text{si p = Profundidad-máxima} \\ \text{máx}_{S_i \in S(n)} f(S_i) & \text{si n es nodo MAX y } p < p_{max} \\ \text{mín}_{S_i \in S(n)} f(S_i) & \text{si n es nodo MIN y } p < p_{max} \end{cases}$$

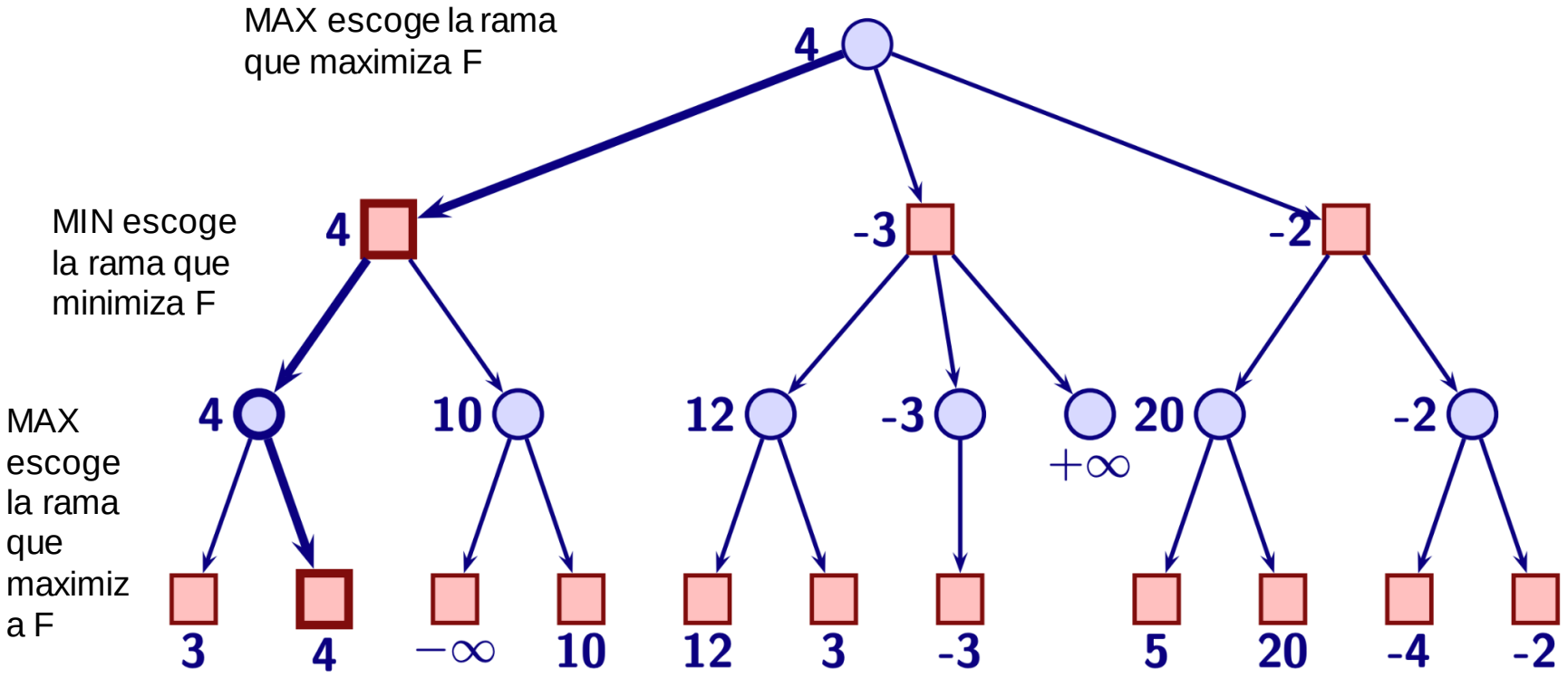
Valores en caso de **estados intermedios**

Ejemplo: nºposibilidades de 3 en raya del jugador - nº posibilidades de 3 en raya del oponente

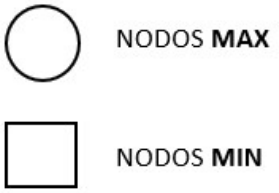
Minimax (informado con f(n))

v_i ○: Nodos MAX Juega MAX (X)

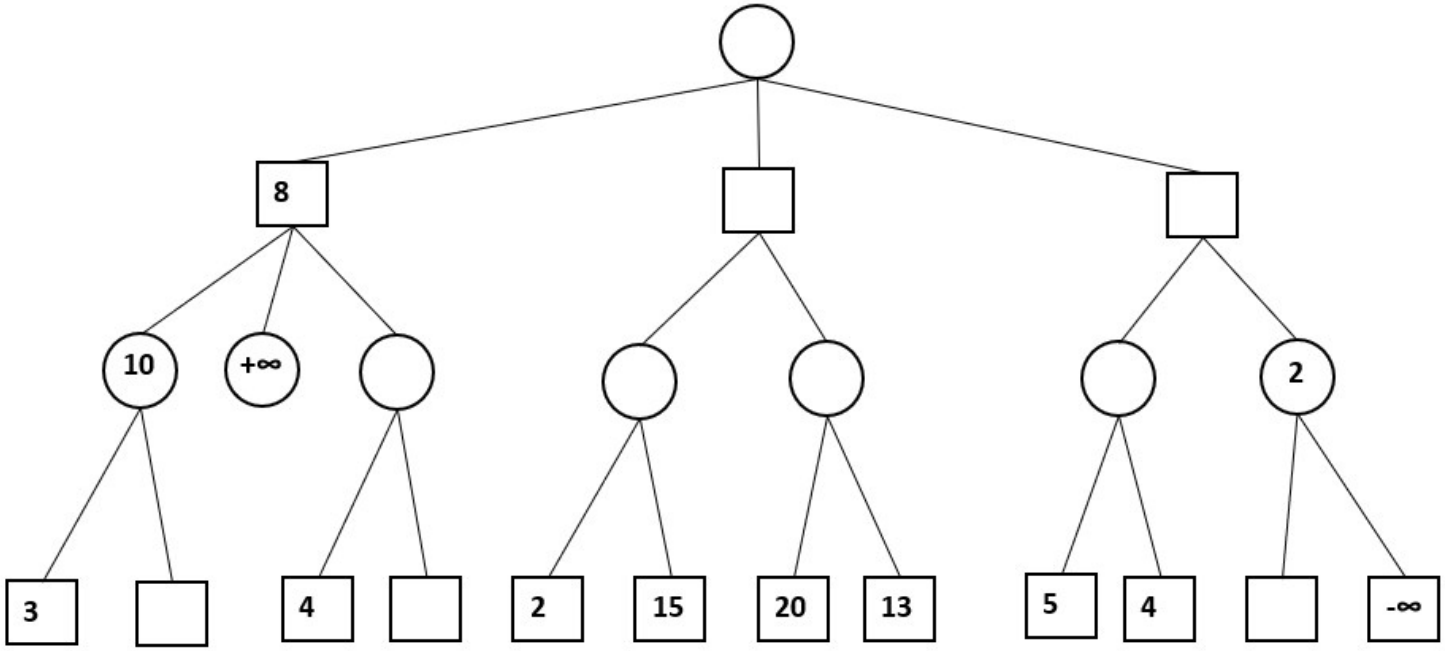
v_i □: Nodos MIN Juega MIN (X)



Ejemplo



Completar los nodos del árbol para que corresponda con una búsqueda Minimax



Propiedades Minimax, ejecutando en profundidad con profundidad limitada

- Es completo
- Es óptimo, contra un oponente óptimo
- Complejidad en tiempo $O(b^d)$ (b: factor de ramificación, d: profundidad máxima)
- Complejidad en espacio $O(b \cdot d)$ (similar a profundidad)

No es necesario tener todos los nodos en memoria: una vez calculado un máximo o un mínimo, se pueden descartar los nodos y hacer backtracking; por eso se implementa de forma similar a profundidad.

Poda alfa-beta

Idea básica

- El algoritmo Minimax se puede seguir descendiendo por una rama hasta la profundidad máxima, y luego retrocediendo.
- Si se hace así, hay situaciones en las que no es necesario calcular el valor de f(n) para ciertos nodos, si se sabe que nunca se escogerán (poda)
- Esto mejora la eficiencia del algoritmo sin variar la solución encontrada. Con poda, algunos nodos no valor definido, ya que justamente lo que se hace es no evaluarlos (ni ellos ni sus descendientes).

Video: primero MINIMAX y luego MINIMAX con poda alfa-beta:
https://www.youtube.com/watch?v=l-hh51ncgDI

Algoritmo MINIMAX (recursivo)

```
SI Profundidad = pmax
    ENTONCES devolver f(Situación)
SI NO SI ganadora (Situación)
    ENTONCES devolver + ∞
    SI NO SI perdedora (Situación)
        ENTONCES devolver - ∞
        SI NO SI empate (Situación)
            ENTONCES devolver 0
            SI NO
                S = sucesores (Situación)
                L = lista de llamadas al MINIMAX
                    (Si ∈ S, Profundidad + 1)
                SI es nivel-max (Profundidad)
                   ENTONCES devolver max (L)
                SI NO devolver min (L)
```

Fuente: OCW UC3M

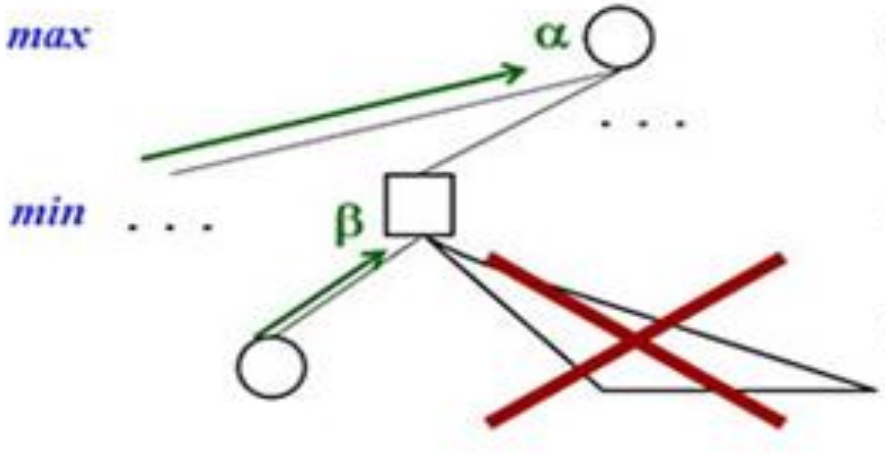
Si estamos en MAX los nodos inferiores los va a evaluar MIN. Si ya tenemos un valor "alfa" en un nodo hermano y una evaluación para un S_i en el nivel siguiente devuelve un valor menor que alfa, ese nodo no se va a escoger nunca y podemos quitarlo

Poda alfa

- Poda α: cualquier rama de un nodo **min** en la que se tenga que α ≥ β se corta (poda) y no se evalúa

1) α es el valor más alto encontrado hasta ahora bajo un nodo max

2) β es el valor más bajo encontrado hasta ahora en los sucesores de un descendiente del anterior, que será un nodo min



Condición de poda: $\alpha \geq \beta$

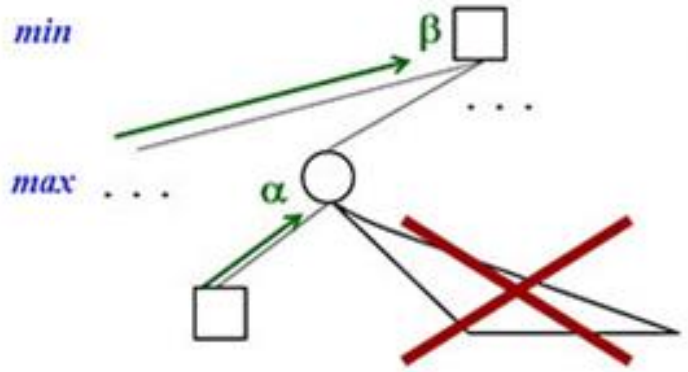
- La evaluación U_{min} del nodo *min* será como mucho β, i.e. $U_{min} \leq \beta$
- La evaluación U_{max} del nodo *max* será $U_{max} = \max(\alpha, U_{min}, \ldots)$
- Si $\alpha \geq \beta$ el valor (exacto) de U_{max} no depende de U_{min}, por lo que no es necesario explorar los sucesores restantes de *min*

Poda beta

- Poda β: cualquier rama de un nodo max en la que se tenga que β ≤ α se corta (poda) y no se evalúa

1) β es el valor más bajo encontrado hasta ahora en los nodos min

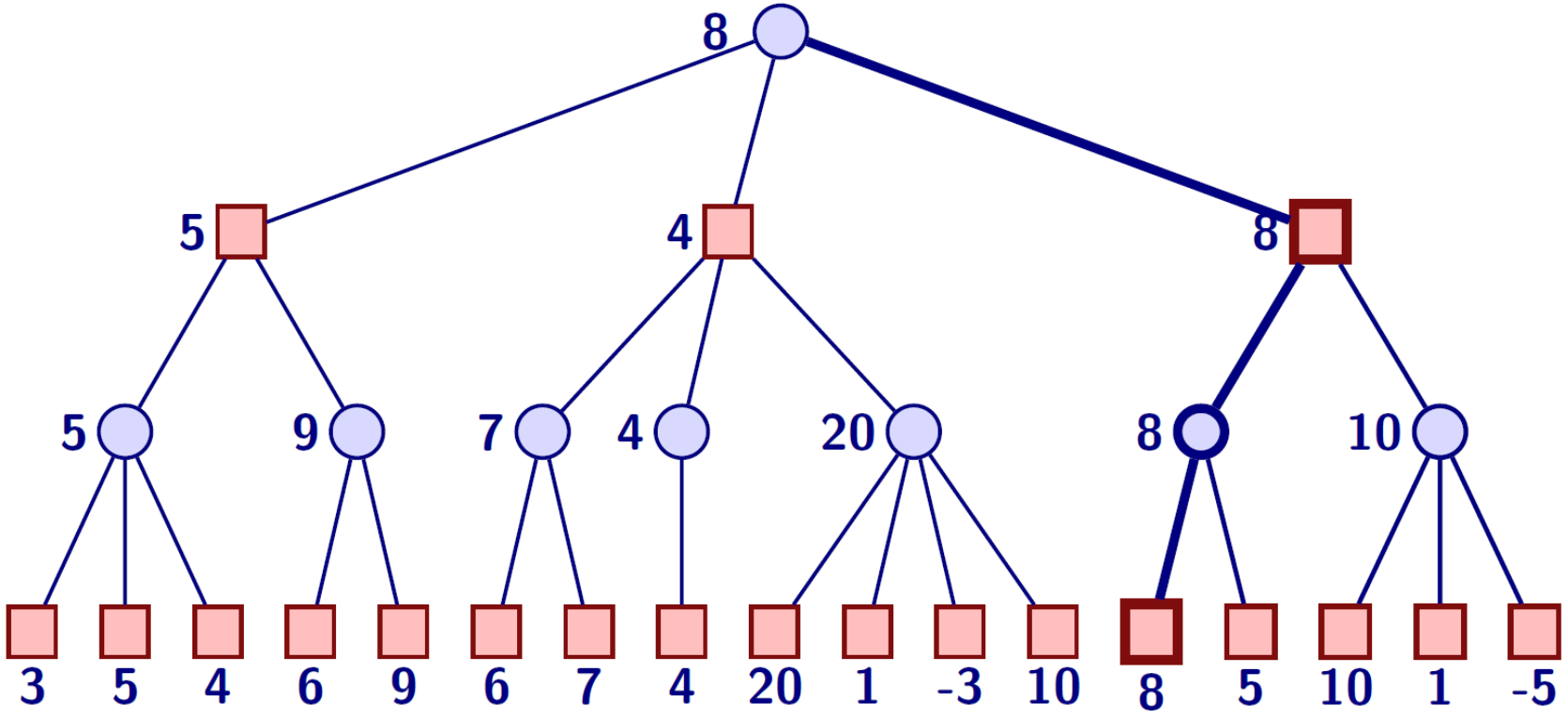
2) α es el valor más alto encontrado en los sucesores de este nodo max



Condición de poda: $\beta \leq \alpha$

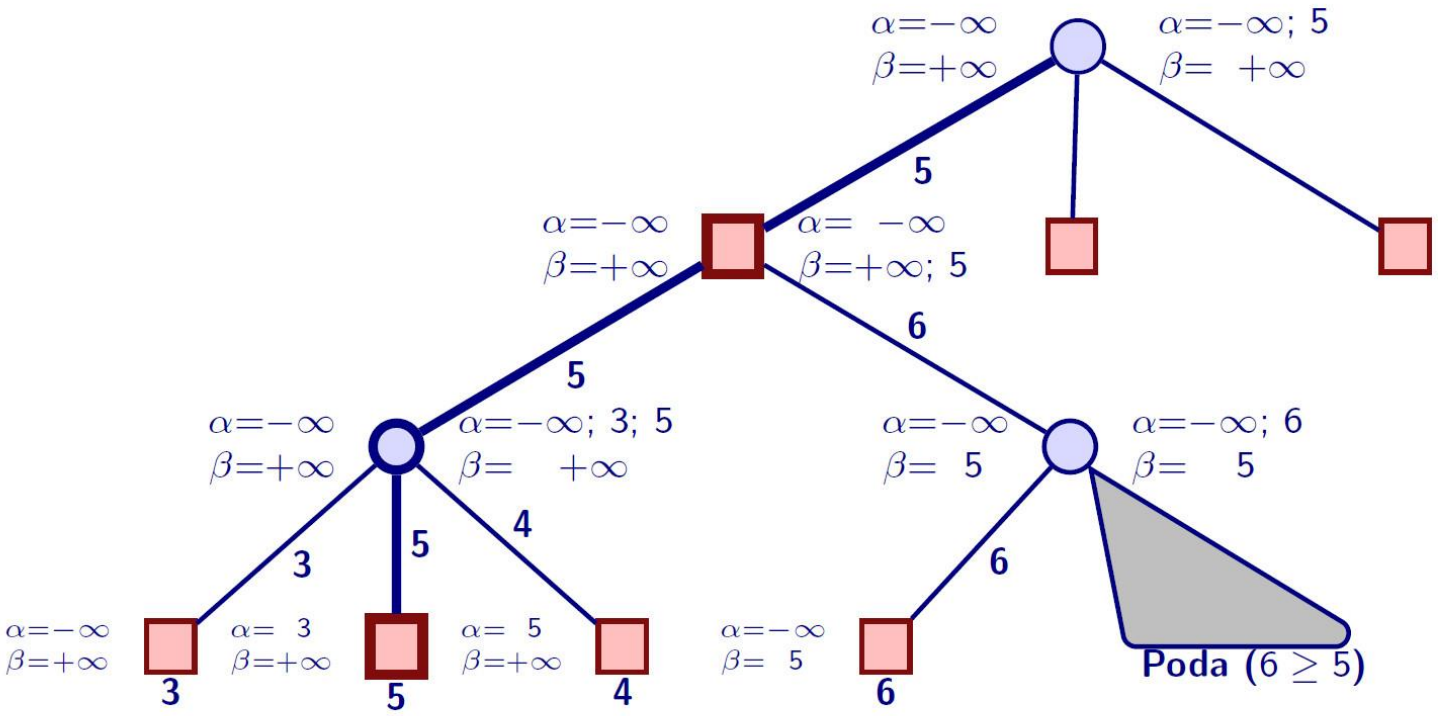
- La evaluación U_{max} del nodo *max* será al menos α, i.e. $U_{max} \geq \alpha$
- La evaluación U_{min} del nodo *min* será $U_{min} = \min(\beta, U_{max}, \ldots)$
- Si $\beta \leq \alpha$ el valor (exacto) de U_{min} no depende de U_{max}, por lo que no es necesario explorar los sucesores restantes de *max*

Cómo funciona

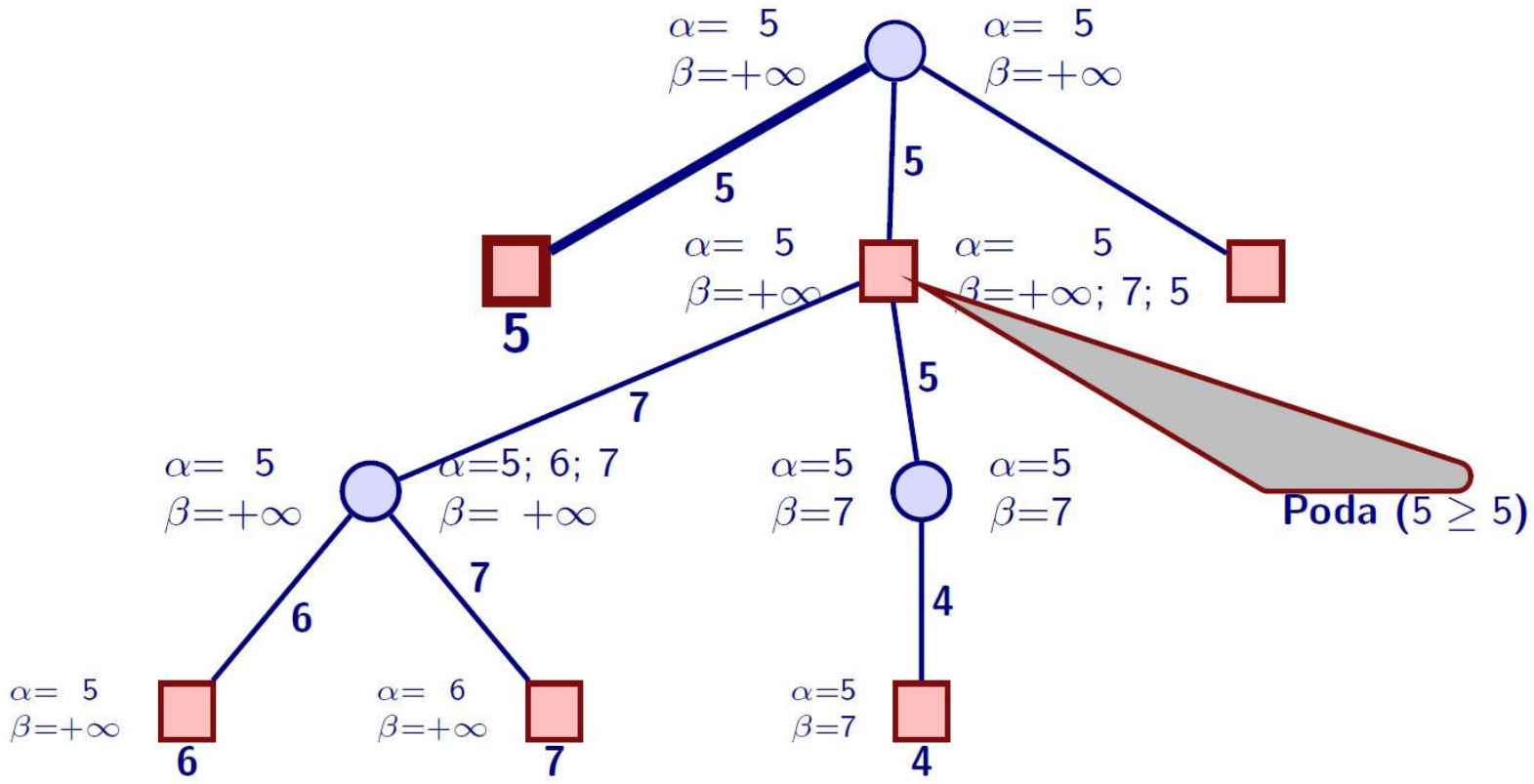


Cómo funciona

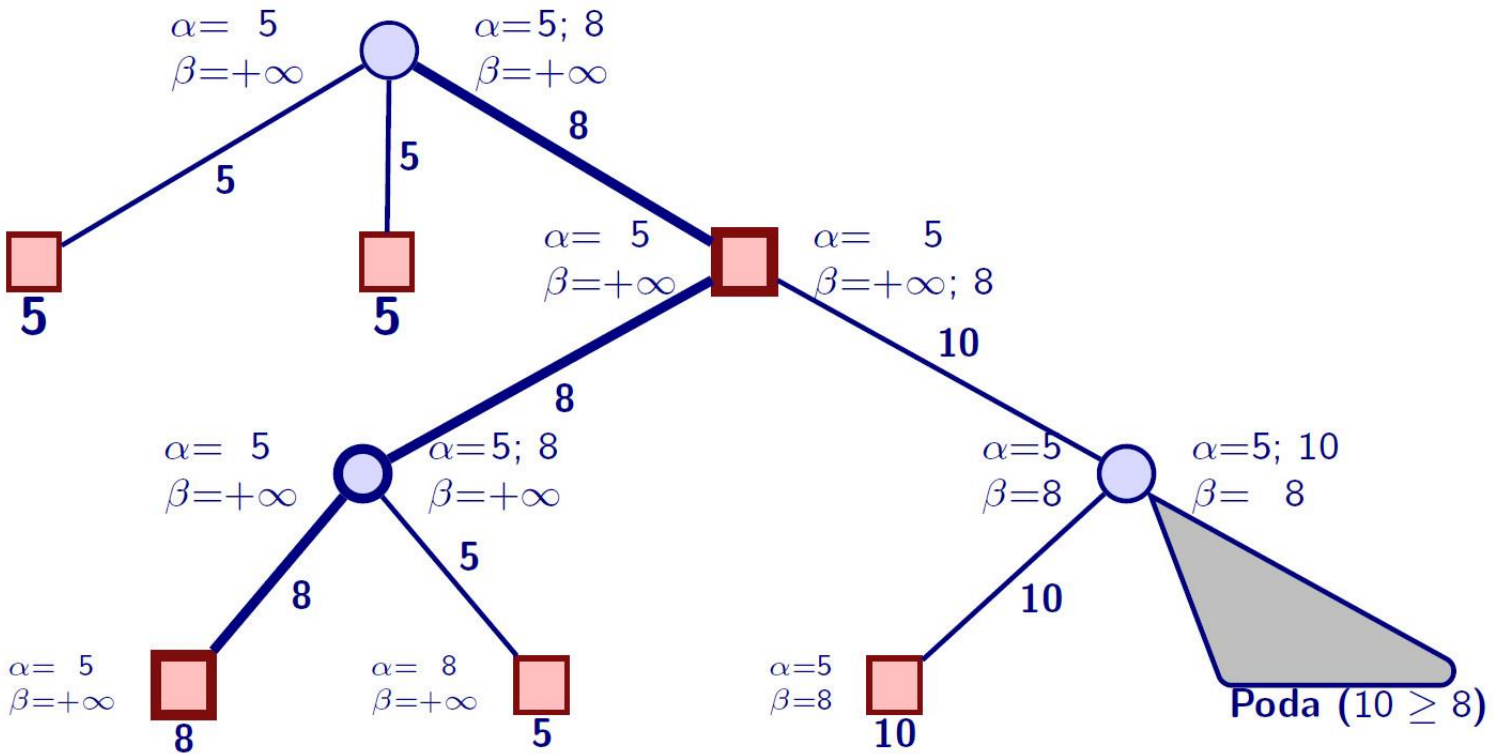
Se recorre por la izquierda.



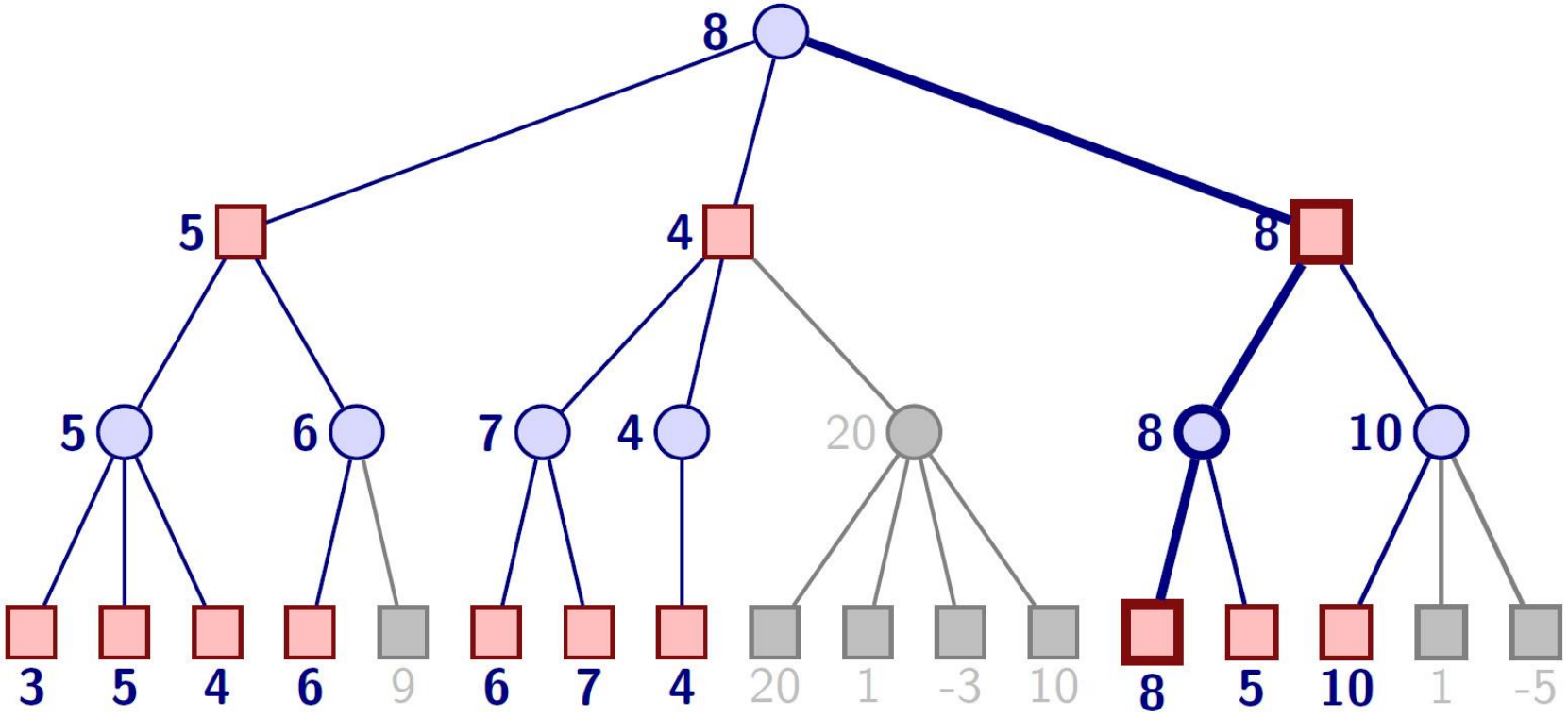
Cómo funciona



Cómo funciona



Cómo funciona



Mejora del rendimiento

- Garantiza la misma solución que Minimax, pero mejora el número de nodos evaluados

Ejemplo de complejidad para factor de ramificación b y profundidad d del árbol		B=10, d=4
Minimax sin poda	$O(b^d)$	10^4= 10 000 nodos
Minimax con poda (mejor caso)	$O(b^{d/2})$	10^2= 100 nodos
Minimax con poda (caso medio)	$O(b^{3\,d/4})$	10^3= 1 000 nodos

Fuente: [Russell & Norvig]

Expectiminimax

Problemas con incertidumbre

- En muchos escenarios existen factores de incertidumbre que determinan el resultado de las acciones o las acciones posibles
- Se modelan insertando nodos azar, donde no se realizan acciones, sino que se escoge un camino en función de una distribución de probabilidad p(r) (r: random)
- Expectiminimax: utilizamos minimax, pero en los nodos Azar añadimos como valor la utilidad ponderada por las probabilidades de cada rama:

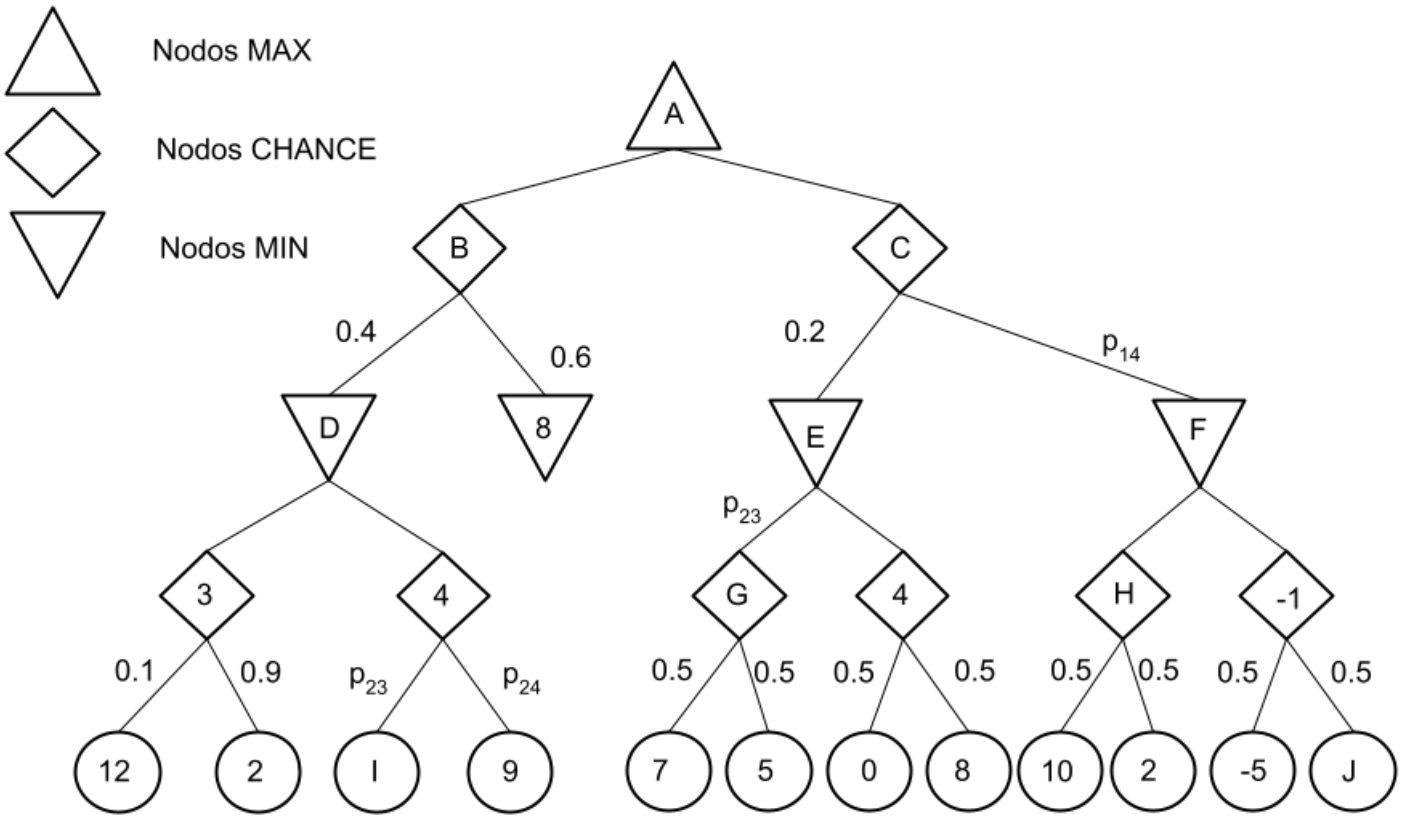
Si cada posible resultado es r_i y lleva al sucesor Si:

$$\text{Valor (azar)} = \text{Suma}_i \left(p(r_i) * \text{Utilidad de } S_i \right)$$

Algoritmo expectiminimax

```
function expectiminimax(node, depth)
    if node is a terminal node or depth = 0
        return the utility value of node
    if the adversary is to play at node
        // Return value of minimum-valued child node
        let α := +∞
        foreach child of node
            α := min(α, expectiminimax(child, depth-1))
    else if we are to play at node
        // Return value of maximum-valued child node
        let α := -∞
        foreach child of node
            α := max(α, expectiminimax(child, depth-1))
    else if random event at node
        // Return weighted average of all child nodes' values
        let α := 0
        foreach child of node
            α := α + (Probability[child] × expectiminimax(child,
depth-1))
    return α
```

Ejercicio expectiminimax



- Completar el árbol con los valores de cada nodo A a H, y los de p_{14}, p_{23} y p_{24}. Nota: el nodo MIN de valor 8 se supone ya calculado a partir de los nodos inferiores.
- ¿Qué decisión debe tomarse en A? ¿Y cómo sigue el resto del camino?

Enlaces de interés

Video sobre minmax, poda alfa-beta (con código)

https://www.youtube.com/watch?v=l-hh51ncgDI

Más completo, con imágenes de Russell & Norvig
(incluye α-β para expectiminimax)

https://www.ics.uci.edu/~dechter/courses/ics-271/fall-12/lecture-notes/04-games.pdf

www.unir.net