Resolución de problemas y espacio de búsqueda

- Métodos informados o heurísticos
  - Introducción
  - Búsqueda primero el mejor
  - Algoritmos de mejora iterativa
  - Búsqueda con adversario
    - Introducción
    - Minimax
    - Poda alfa-beta
    - Consideraciones prácticas

Búsqueda con adversario

- Búsqueda en un entorno hostil, competitivo, impredecible
  - Adversario(s) contrario(s) a nuestros objetivos
- A los problemas de búsqueda con adversario (conflicto de intereses) se les suele denominar juegos
- De cara a simplificar, consideraremos principalmente juegos con las siguientes características:
  - 2 jugadores cuyas jugadas se alternan
  - Al acabar, cada jugador pierde, gana o empata
    - Juegos de suma cero (o nula)
      - Lo que “gana” uno es lo que “pierde” el otro
  - Totalmente observables, todo a la vista
    - Información perfecta (o completa)
      - No interviene el azar

Búsqueda con adversario

- Un árbol de juego es una representación explícita de todas las secuencias de jugadas posibles en una partida
  - Cada nivel representa, alternativamente, las movimientos posibles de cada jugador
  - Las hojas corresponden a estados GANAR, PERDER o EMPATAR
  - Cada camino desde la raíz (el estado inicial del juego) hasta una hoja representa una partida completa
  - El espacio de estados se suele representar como un árbol
- Los algoritmos de búsqueda vistos hasta ahora no sirven
- El problema ya no es encontrar un camino en el árbol de juego (puesto que esto depende de los movimientos futuros que hará el oponente), sino decidir el mejor movimiento dado el estado actual del juego
  - Se podría usar escalada para elegir el siguiente movimiento
    - Lo limitaría a un solo nivel

Minimax

- Se suele denominar MAX al jugador que mueve primero, y al otro MIN
- Son nodos MAX (MIN) aquéllos en los que tiene que jugar MAX (MIN)
  - Si identificamos la raíz con el nivel 0, y comienza jugando MAX, los nodos de nivel par le corresponden a MAX y los de nivel impar a MIN
- En el árbol de juego completo se asigna a los nodos terminales un valor de 1, -1 o 0 según si gana MAX (G), gana MIN (P) o empatan (E)
- A partir de los valores asignados a los nodos terminales, podemos "ascender" los valores hasta la raíz *(propagar hacia arriba)*:
  - A cada nodo MAX se le asigna el máximo de los valores de sus hijos
    - MAX intenta maximizar su ventaja
      - Buscamos el "mejor" movimiento para MAX
  - A cada nodo MIN se le asigna el mínimo de los valores de sus hijos
    - MIN procura minimizar la puntuación de MAX
      - Asumimos que siempre intentará elegir el "peor" movimiento para MAX, es decir, el "mejor" para sus intereses *(siempre jugará óptimamente)*

Minimax

- El valor ascendido a la raíz indica el valor del mejor estado que el jugador MAX puede aspirar a alcanzar
  - La etiqueta de un nodo nos indica lo mejor que podría jugar MAX en el caso de que se enfrentase a un oponente perfecto
- Resolver un árbol de juego significa asignar una etiqueta a la raíz con el método anterior
- Un árbol solución (o estrategia de juego) para MAX es un subárbol del árbol de juego que
  - contiene a la raíz
  - contiene un sucesor de cada nodo MAX no terminal que aparezca en él
  - contiene todos los sucesores de cada nodo MIN que aparezca en él

Minimax

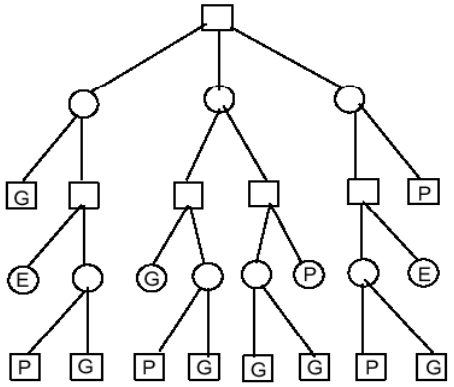
- Un árbol solución representa un plan (estrategia) de los movimientos
  - que debe realizar MAX ante cualquier movimiento posible de MIN
- Con un árbol solución para MAX,
  - se diseña un programa que juegue con un contrincante MIN (humano o máquina).
  - No asegura que vaya a ganar
- Un árbol solución para MAX
  - se llamará árbol ganador para MAX (o estrategia ganadora para MAX)

    si todos los nodos terminales del árbol solución tienen etiqueta 1 (G)
  - Un árbol ganador para MAX asegura que

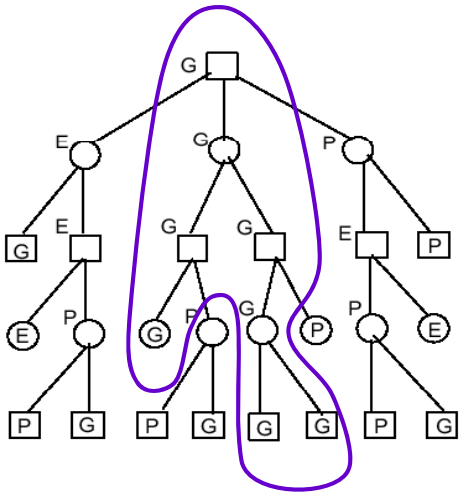
    el jugador MAX ganará, haga lo que haga MIN
- Existirá un árbol ganador para MAX si y sólo si
  - al resolver el árbol de juego la etiqueta de la posición inicial es 1 (G)

Ejemplo

Árbol de juego sin resolver



Árbol de juego resuelto



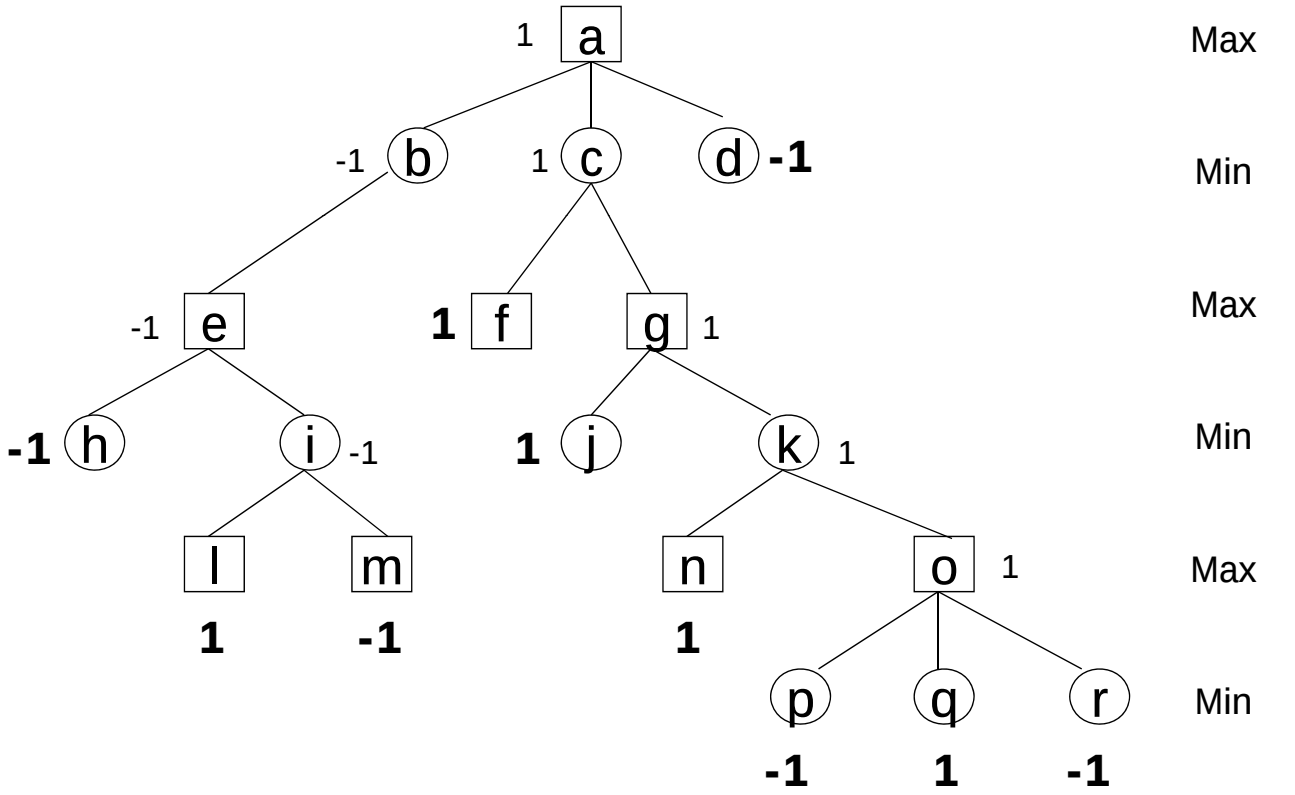
G = 1, E = 0, P = -1
Nodos MAX: cuadrados
Nodos MIN: círculos

Árbol ganador para MAX

Minimax

- El árbol de juego debe ser generado dinámicamente
  - No es necesario mantener todo el árbol en memoria *(generación 1º en profundidad)*
- El método de etiquetado descrito requiere un árbol de juego completo
  - En la práctica, para la mayoría de los juegos, desarrollar todo el árbol de juego es una tarea impracticable
    - Muchos posibles movimientos en cada paso hacen que el árbol crezca enormemente
  - Damas $\approx 10^{40}$ nodos no terminales
    - Generar el árbol completo requeriría 10^{21} siglos (3 billones nodos/seg.)
  - Ajedrez $\approx$ unos 10^{120} nodos y unos 10^{101} siglos
    - Incluso si un adivino nos proporcionase un árbol ganador sería imposible almacenarlo o recorrerlo
- Minimax es exponencial en tiempo por lo que sólo resulta aplicable directamente en juegos muy simples *(árboles de juego manejables)*

Ejemplo



--- Minimax con estimación en nodos límite ---

- Aproximación práctica: generar el árbol de juego sólo hasta un cierto nivel límite *(horizonte limitado)*
  - El que permitan los recursos disponibles (tiempo y/o espacio)
  - Cuanto más profundo sea el límite, mayor será el horizonte
- Las "hojas" de ese subárbol no se corresponden con finales de partida
  - No se les puede asignar un valor que refleje si llevarán a ganar o no
- Aproximación heurística: se asigna un valor a esos nodos "hoja" del subárbol de juego según alguna función heurística
  - Estimación heurística de la "bondad" del estado correspondiente (tendencia a ganar, perder o empatar)
    - Nodos terminales (finales de partida) ➔ función de evaluación
    - Nodos límite (nivel de exploración) ➔ función de estimación
      - Valores positivos grandes a los nodos más favorables para MAX
  - Ése es el valor que se propaga hacia arriba

Minimax con estimación en nodos límite

- Se asume que el valor ascendido hasta la raíz, obtenido mediante una profundización hasta un cierto límite *n*, va a ser una estimación mejor que si sólo aplicáramos directamente la función de estimación a los sucesores del nodo raíz
  - Una estrategia de tipo escalada no tendría en cuenta la secuencia de futuras respuestas del oponente y dependería demasiado de la calidad de la función de estimación
- A la raíz le llega la medida heurística del mejor estado alcanzable en *n* movimientos
  - Cuanto mayor sea el horizonte, más seguros son los elementos de decisión para elegir la mejor jugada
  - Así se preven las consecuencias de la jugada a más largo plazo
  - Pero ese "mirar hacia delante limitado" no ofrece garantías

Implementación de minimax

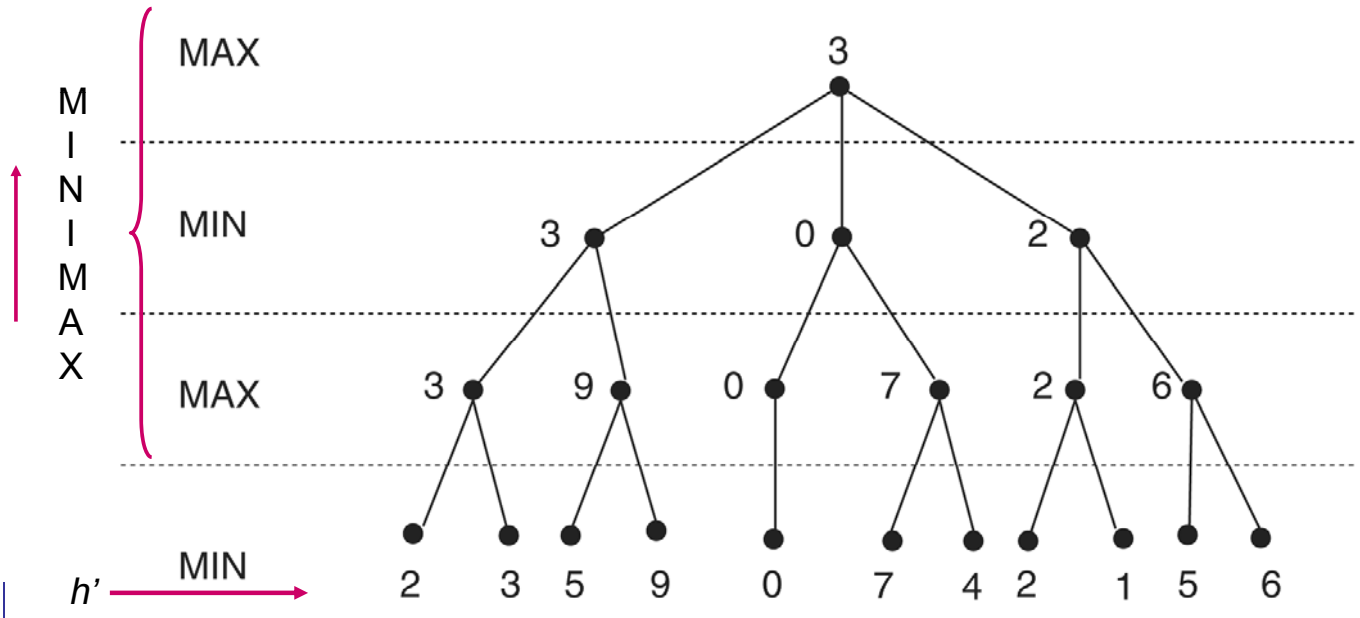
```
/* Llamada inicial: MAX_VALOR (estado, límite) */

función MAX_VALOR (estado, prof) devuelve valor      del estado
    si prof = 0 entonces
        devolver estimar(estado)
    si estado es nodo terminal entonces
        devolver evaluar(estado)
    si no
        valor = -∞
        para cada SUCESOR s de estado hacer
            valor = maximo(valor, MIN_VALOR(s, prof-1))
        devolver valor

/* función MIN_VALOR análoga */
```

Ejemplo: valor minimax de 3 capas

Cálculo del valor minimax mirando hacia delante 3 capas (niveles)



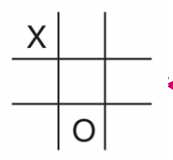
Ejemplo: función h' de estimación para el tres en raya

Heurística: intentar medir el conflicto del juego $h'(n) = M(n) - O(n)$

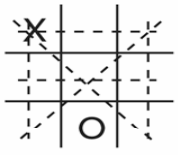
- $M(n)$ = nº de mis posibles líneas ganadoras *(contando vacías)*
- $O(n)$ = nº de sus posibles líneas ganadoras *(contando vacías)*

- $+\infty$ si final: MAX gana
- $-\infty$ si final: MIN gana

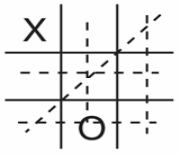
Para evitar confusión con los valores de *h'*, se distinguen los nodos finales de partida con valores fuera del rango de *h'*



X tiene 6 posibles líneas ganadoras

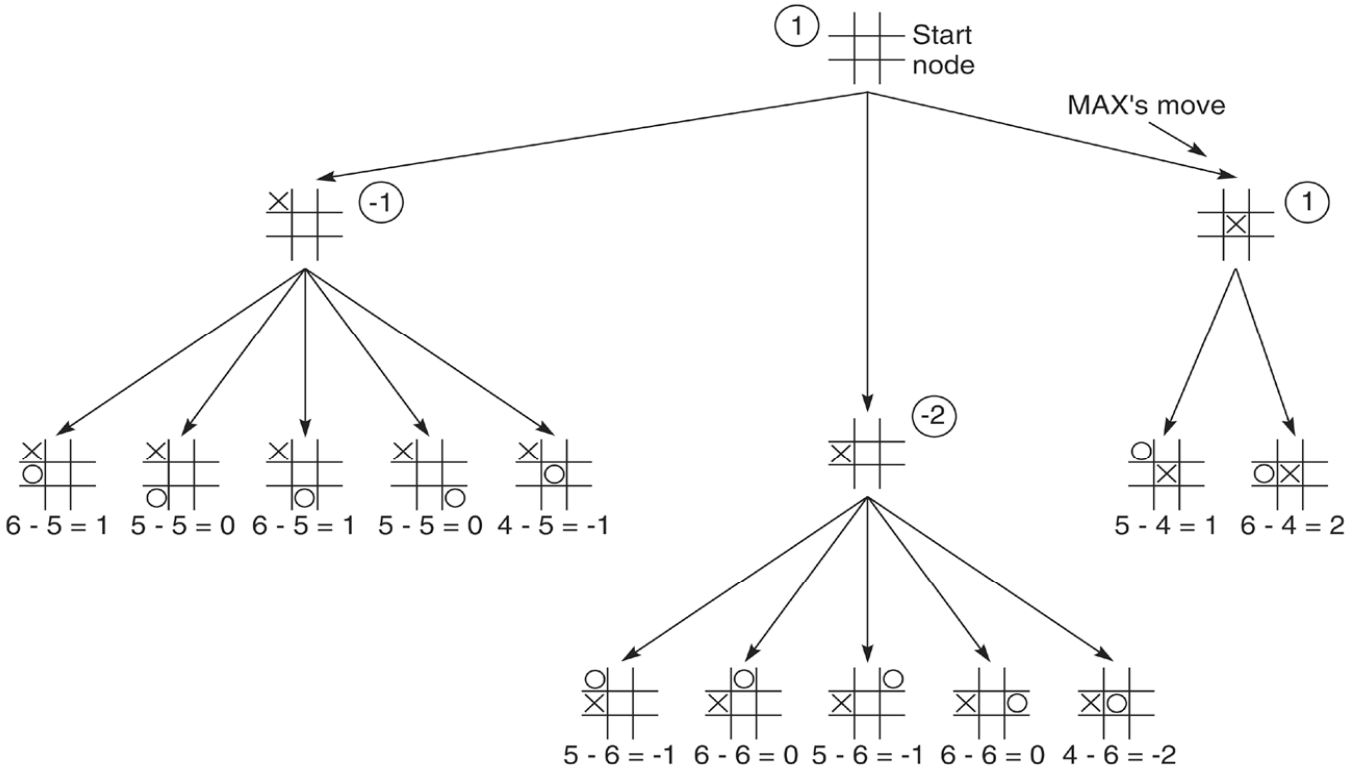


O tiene 5 posibles líneas ganadoras

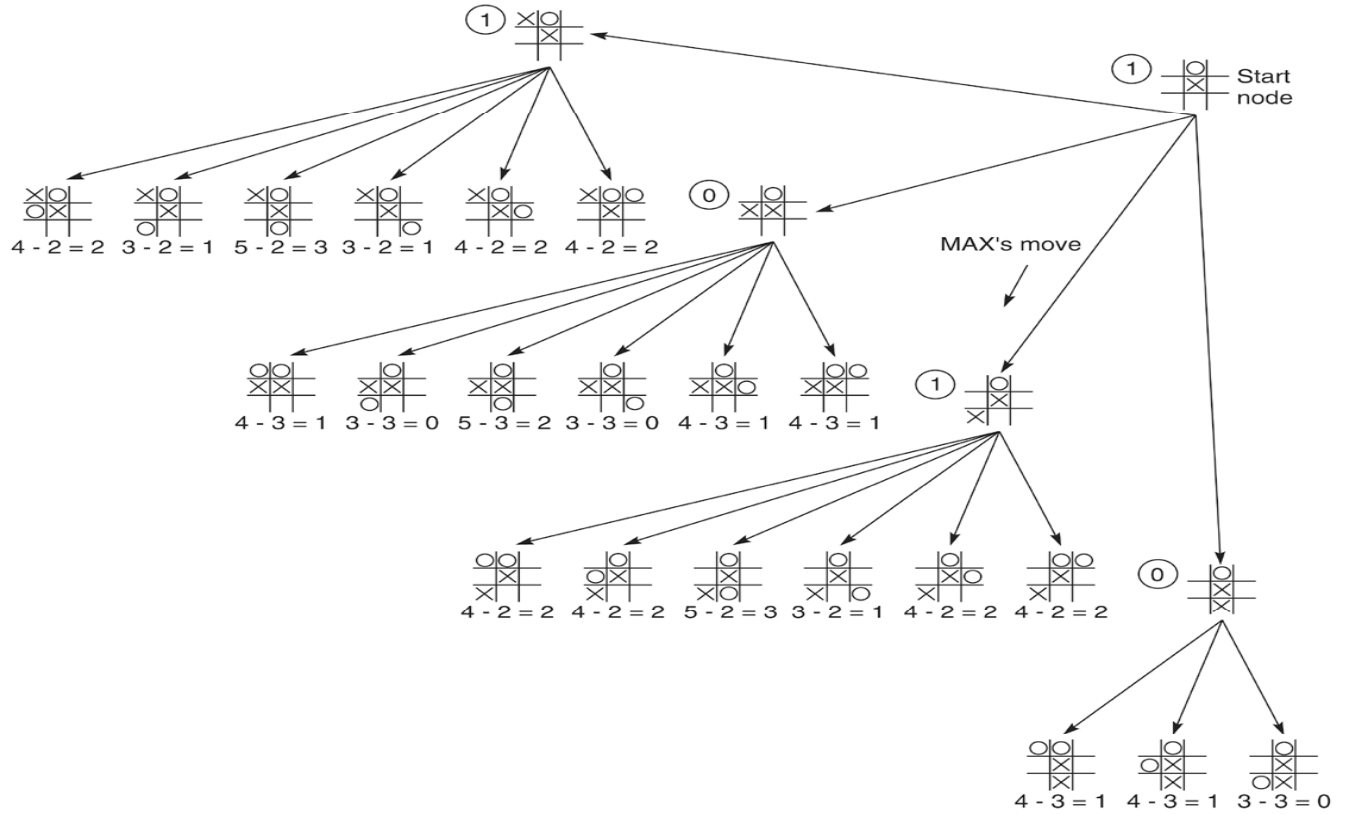


$h'(n) = 6 - 5 = 1$

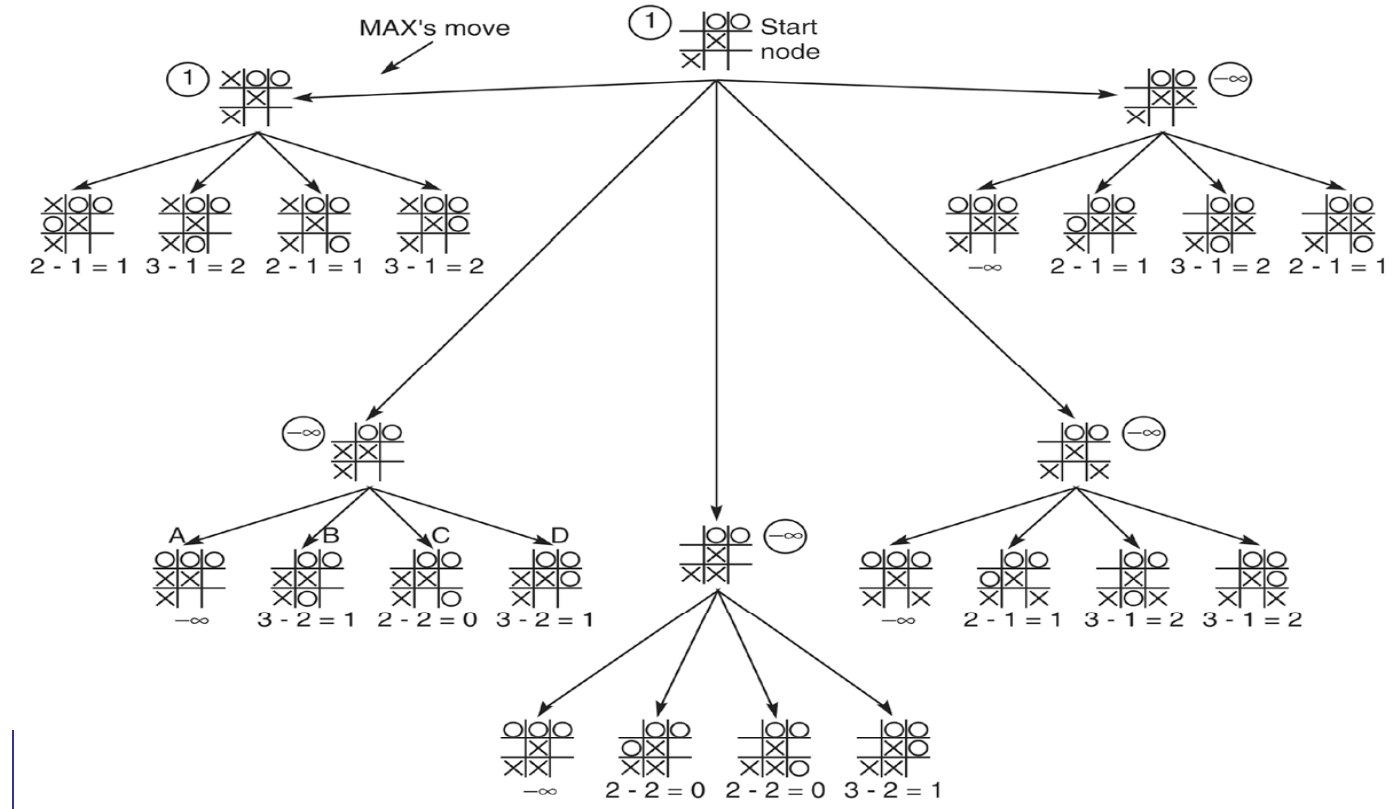
Movimiento inicial mirando 2 capas por delante



Segundo movimiento de MAX (2 capas)



Movimiento de MAX cercano al final (2 capas)



La función de estimación

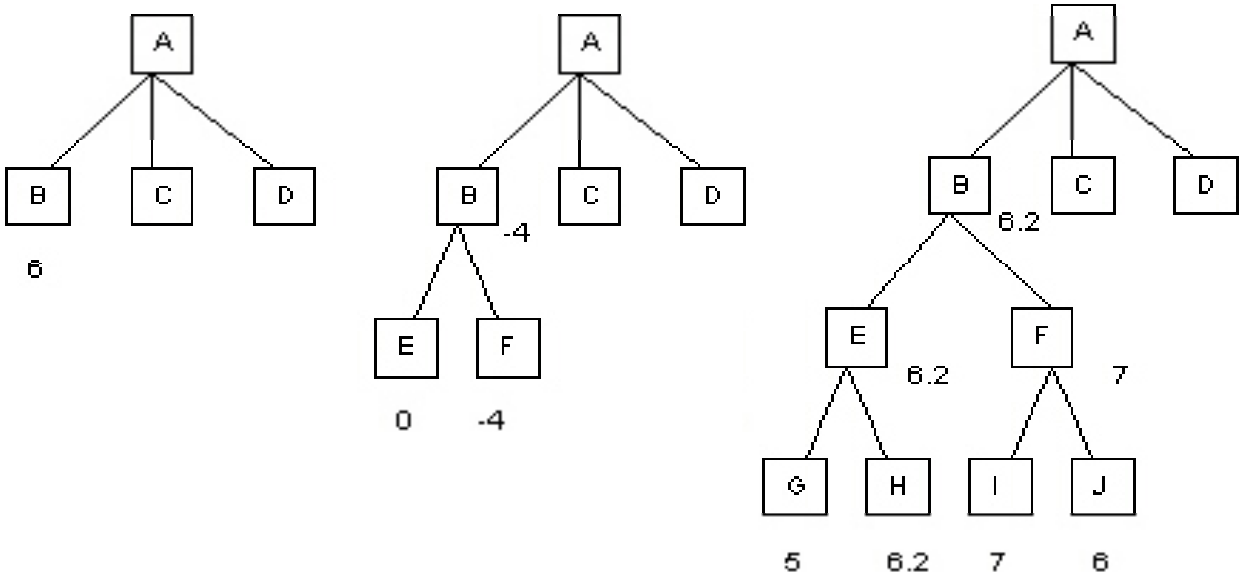
- En los nodos terminales (de final de partida), el valor de la función de estimación debe coincidir con el de la función de evaluación
- La función de estimación *(heurística)* suele tener en cuenta distintas características del estado del juego ponderadas por distintos pesos:
  - Número de piezas propias y del contrario (ventaja de uno sobre el otro)
  - Ponderación de la importancia de las piezas en ajedrez
  - Posiciones de las piezas en el tablero
  - Puntos débiles (peón aislado, etc.)
  - Características posicionales (protección del rey, capacidad de maniobra, control del centro del tablero, etc.)
- Cuanto más tiempo se gaste calculando la función de estimación,
  - menos tiempo se puede dedicar a la búsqueda
  - Hay que llegar a un compromiso

    entre la calidad de la estimación y su coste

La profundidad límite de exploración: Mejoras

- Lo más sencillo es realizar búsquedas hasta una profundidad fija (la máxima que nos permitan nuestros recursos)
  - Pero puede darse el efecto horizonte: se evalúa como bueno o malo un nodo sin saber que en la siguiente jugada la situación se revierte
- Resulta más efectivo elegir una profundidad menor, explorar todas las ramas hasta esa profundidad y dedicar los recursos ahorrados a profundizar más en ciertas ramas con:
- Búsquedas secundarias:
  1.- Búsqueda de la quietud (o espera del reposo)
    - Continuar la búsqueda hasta alcanzar una situación estable *(aparentemente)*
  2.- Extensiones singulares
    - Si un nodo hoja es muy diferente a sus hermanos, el nodo se expande una capa más por si se da una situación de captura inminente (jugada forzada)
  3.- Otros tipos
    - Movimientos de libro (aperturas y finales de partida)...

Búsqueda de la quietud

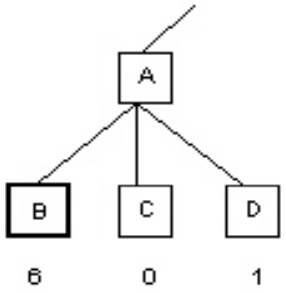


Explorando un nivel la estimación de B es 6

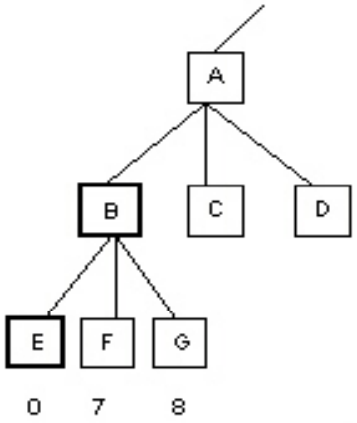
Se explora 1 nivel más y hay un cambio drástico

Reposo alcanzado: estimación similar a la inicial

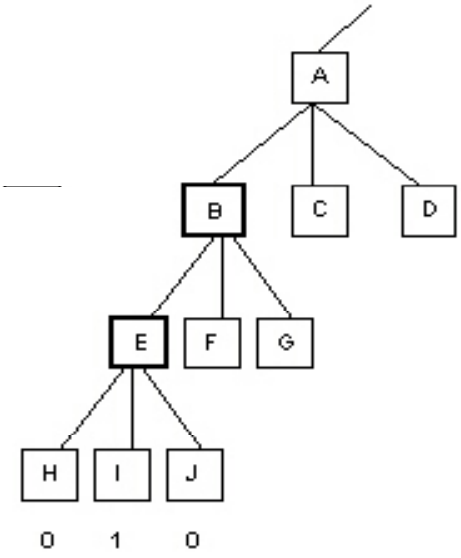
Extensiones singulares



B tiene un valor superior al resto de sus hermanos. ¿Captura inminente?



¿Captura por parte del oponente?



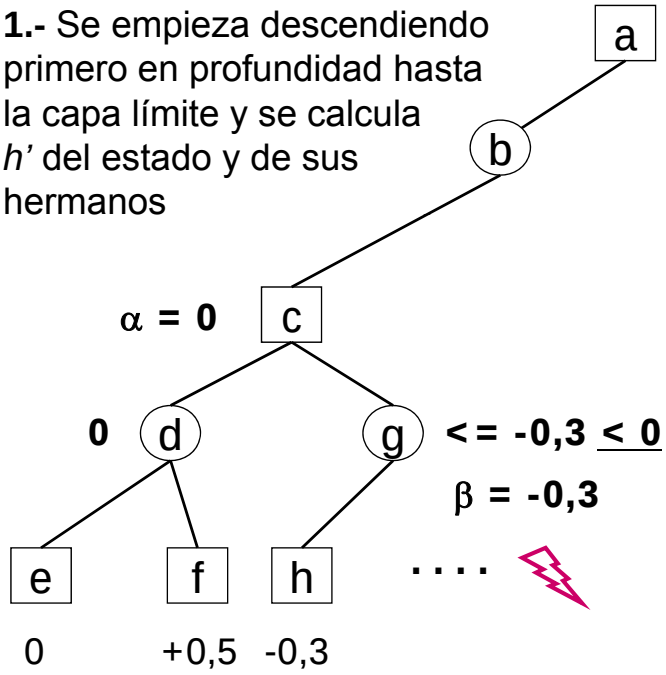
No hay captura: valores en un intervalo estrecho. Se interrumpe la búsqueda secundaria

---- Poda alfa-beta ----

- Optimización de minimax, que genera el espacio de búsqueda entero y después evalúa y propaga (recorrido doble)
  - Un solo recorrido: generación y evaluación simultáneas
  - Poda: algunas ramas del árbol no necesitan ser analizadas
    - Abandono de soluciones parciales por ser peores que otras
- Alfa-beta realiza una búsqueda de tipo primero en profundidad
- Para cada nodo MAX se calcula un valor α que representa el valor máximo de los sucesores de MAX generados hasta ese momento
  - El valor α representa una cota inferior para el valor final que pueda alcanzar el nodo MAX (lo peor que le podría ir a MAX)
    - Se inicializa a $-\infty$ y nunca puede decrecer
- Para cada nodo MIN se calcula un valor β que representa el valor mínimo de los sucesores de MIN generados hasta ese momento
  - El valor β representa una cota superior para el valor final que pueda alcanzar el nodo MIN (lo mejor que le podría ir a MIN)
    - Se inicializa a $+\infty$ y nunca puede crecer

Poda alfa

1.- Se empieza descendiendo primero en profundidad hasta la capa límite y se calcula *h'* del estado y de sus hermanos



¡MAX nunca elegirá g!

2.- Si son nodos MAX, el mínimo va a al padre y este valor se ofrece al abuelo como valor provisional α
Sigue descendiendo a otros nietos,pero

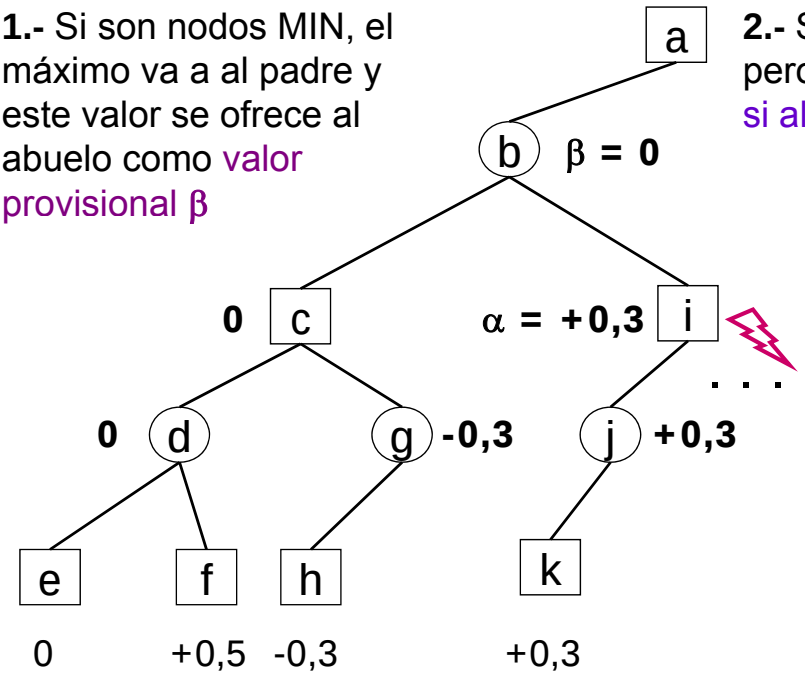
3.- termina la exploración del padre si alguno de sus valores es <= α

Se puede cortar la búsqueda por debajo de cualquier nodo MIN cuyo valor β sea menor o igual que el valor α de algún antecesor MAX

A ese nodo MIN, (**g**), se le asigna definitivamente su valor β que puede no coincidir con el que habría obtenido minimax *(podría ser <)* pero esto no afecta a la elección de movimiento

Poda beta

1.- Si son nodos MIN, el máximo va a al padre y este valor se ofrece al abuelo como valor provisional β



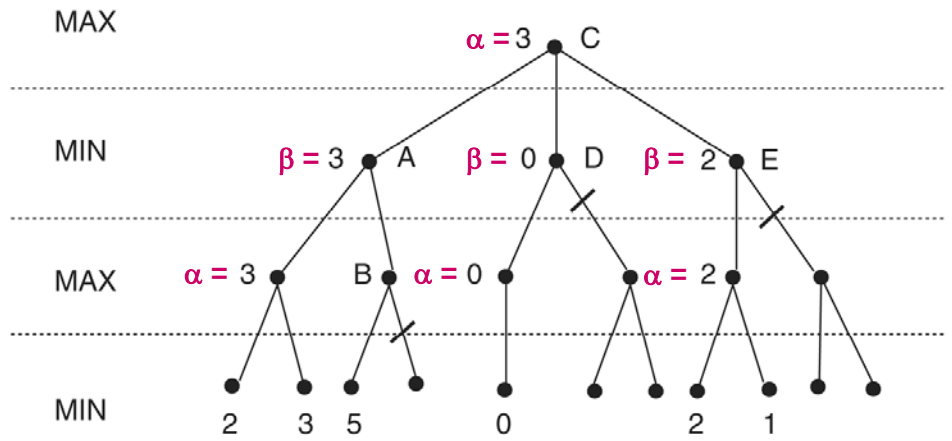
¡MIN nunca elegirá i!

2.- Sigue descendiendo a otros nietos, pero termina la exploración del padre si alguno de sus valores es >= β

Se puede cortar la búsqueda por debajo de cualquier nodo MAX cuyo valor α sea mayor o igual que el valor β de algún antecesor MIN

A ese nodo MAX, (**i**), se le asigna definitivamente su valor α que puede no coincidir con el que habría obtenido minimax *(podría ser >)* pero esto no afecta a la elección de movimiento

Ejemplo



A tiene β = 3 (A <= 3)

B es β-podado (B >= 5 y 5 > 3)

C tiene α = 3 (C >= 3)

D es α-podado (D <= 0 y 0 < 3)

E es α-podado (E <= 2 y 2 < 3)

C tiene valor 3 definitivo (valor minimax)

Alfa-beta (inicialización y poda beta)

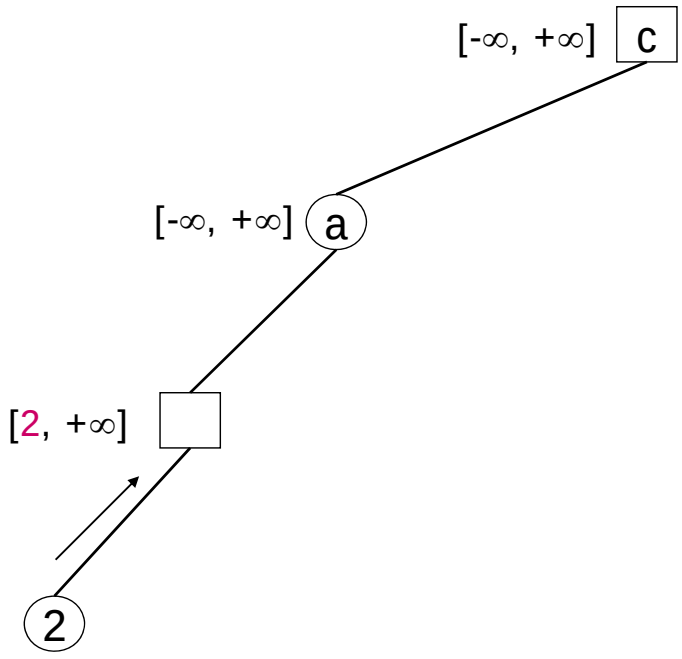
```
función BÚSQUEDA_ALFA_BETA (estado) devuelve valor
    {permite seleccionar un movimiento con ese valor}
      valor = MAX_VALOR (estado, -∞, +∞) {estado es un nodo MAX}
      devolver valor

función MAX_VALOR (estado, alfa, beta) devuelve valor
    si estado es un nodo terminal entonces
          devolver evaluar(estado)
    si no
          valor = -∞ {antes de empezar a descender por el primer hijo}
          para cada SUCESOR s de estado hacer
                valor = maximo(valor, MIN_VALOR(s, alfa, beta))
                si valor >= beta entonces devolver valor {poda}
                si no alfa = maximo(alfa, valor)
          devolver valor
```

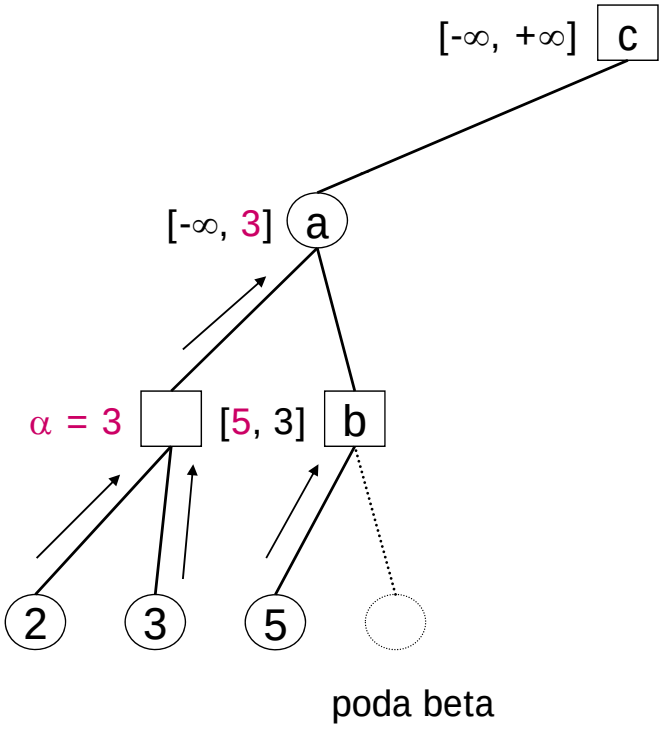
Alfa-beta (poda alfa)

```
función MIN_VALOR (estado, alfa, beta) devuelve valor
    si estado es un nodo terminal entonces
        devolver evaluar(estado)
    si no
        valor = +∞ {antes de empezar a descender por el primer hijo}
        para cada SUCESOR s de estado hacer
            valor = minimo(valor, MAX_VALOR(s, alfa, beta))
            si valor <= alfa entonces devolver valor {poda}
            si no beta = minimo(beta, valor)
        devolver valor
```

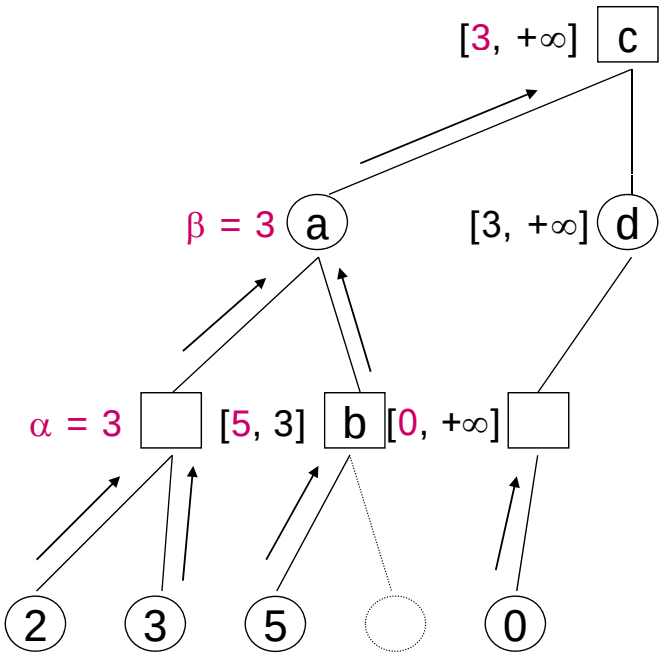

Ejemplo paso a paso



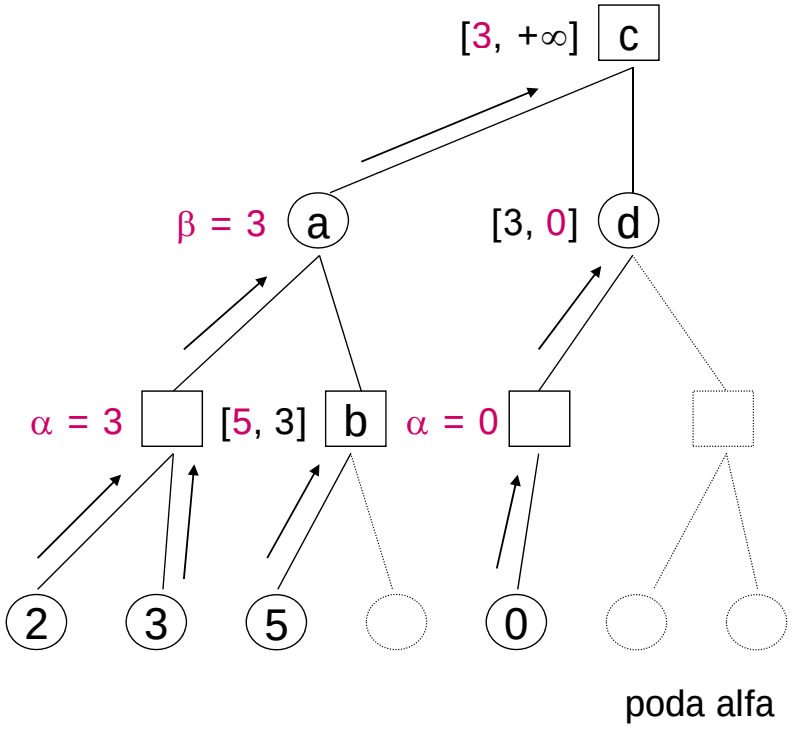
Ejemplo paso a paso



Ejemplo paso a paso

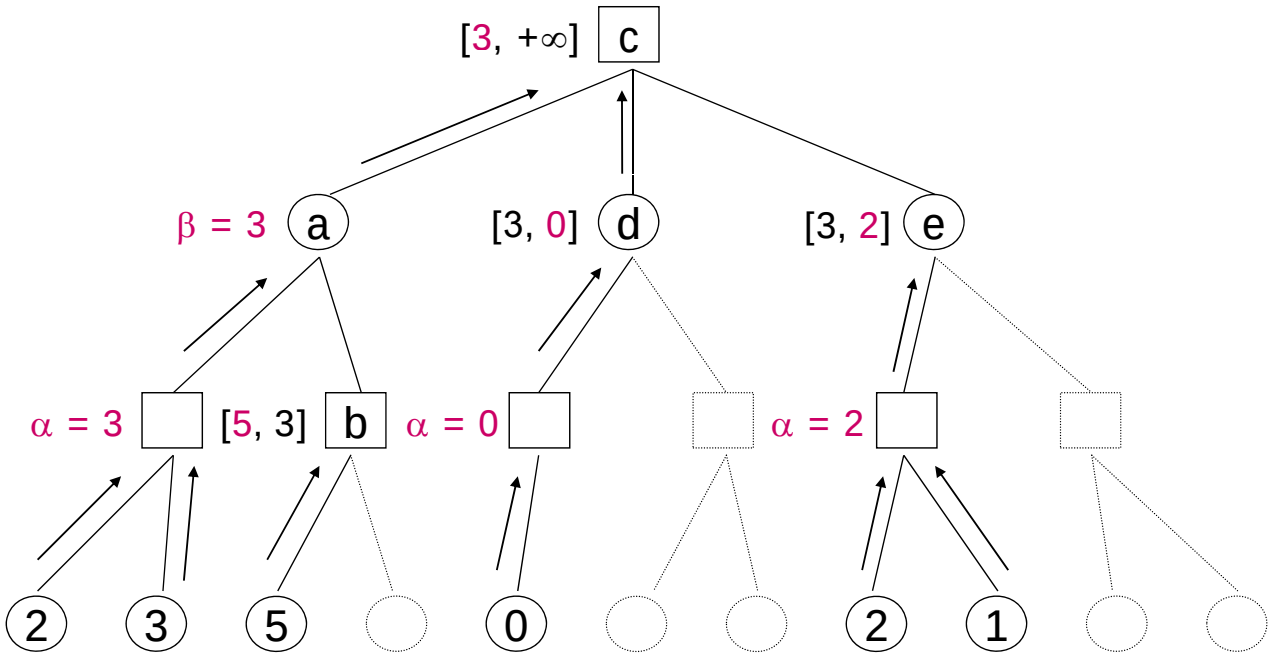


Ejemplo paso a paso



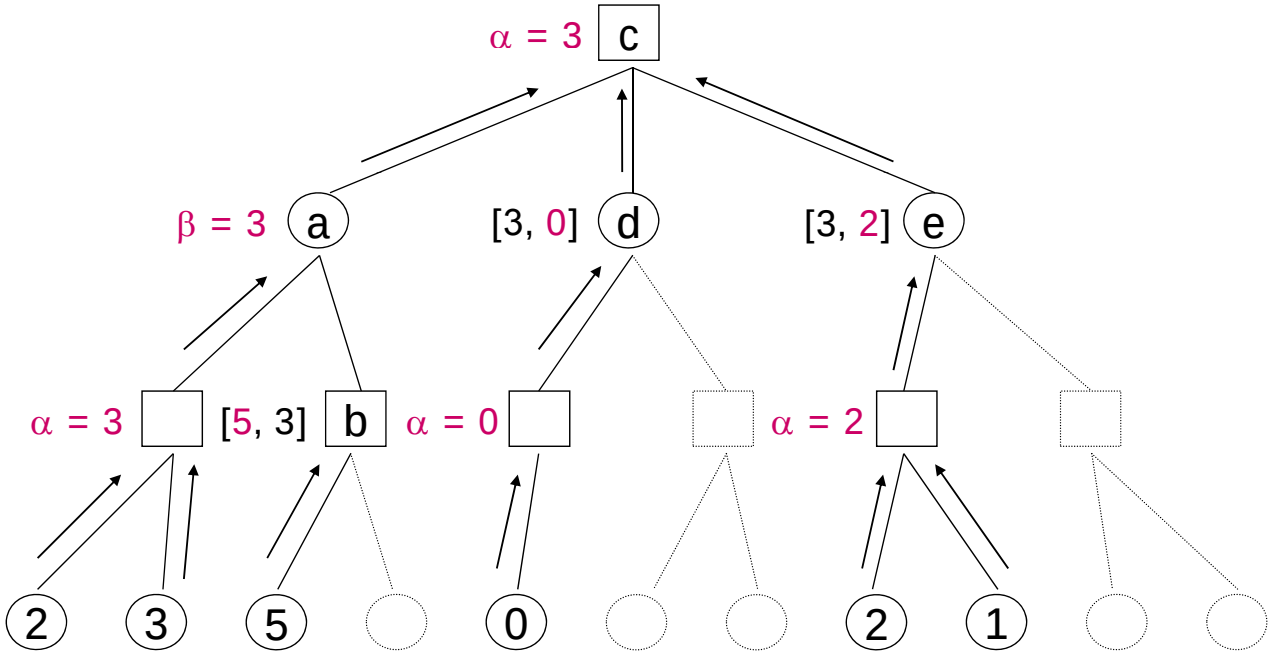
poda alfa

Ejemplo paso a paso



poda alfa

Ejemplo paso a paso

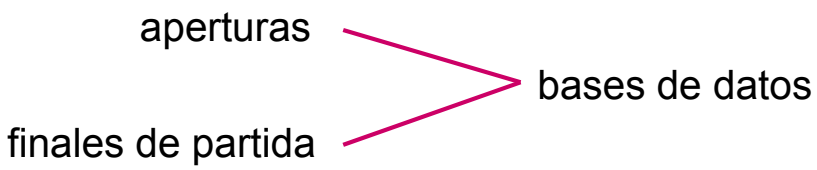


Propiedades de la poda alfa-beta

- Este algoritmo es de tipo ramificación y poda *(branch & bound)*
- Garantiza encontrar el mismo mejor movimiento (u otro equivalente con el mismo valor) que minimax, pero de forma más eficiente
- Si el mejor nodo límite es el generado primero, las podas serán máximas y habrá que generar y evaluar el mínimo número de nodos
  - Influye mucho el orden de generación de los sucesores
  - En esas condiciones, alfa-beta tiene que examinar sólo $O(r^{p/2})$ nodos para escoger el mejor movimiento, en vez de $O(r^p)$ con minimax
    - El factor de ramificación efectivo sería $\sqrt{r}$ en lugar de r
    - Permite profundizar aproximadamente el doble (= tiempo)
- Si pudiera utilizarse una función de ordenación en la generación de nodos, se garantizaría *(si la tuviéramos, jugaríamos perfecto...)*
  - Para un nodo MAX debería generar primero el hijo de mayor valor
  - Para un nodo MIN el de menor valor
  - Al menos podemos usar la función de estimación...

Ajedrez: consideraciones prácticas

	Tiempo
1. Generación de movimientos (ordenados por *h'*)	50%
❑ para máximo aprovechamiento de la poda alfa-beta	
2. Evaluación estática	40%
❑ función de estimación a las hojas	
3. Búsqueda	¡10%!



- ❑ Todas estas ideas estaban ya claras a finales de los 60
  - ❑ Mejoras por incremento en la potencia de cómputo de los ordenadores
  - ❑ Y por unas pocas nuevas ideas

Ajedrez: consideraciones prácticas

- ❑ Factor de ramificación altamente variable (poda alfa-beta)
  - ❑ Con límite fijo en la exploración, unas veces va rápido, otras muy lento
- ❑ Se usa profundización iterativa
  - ❑ Así siempre se tiene disponible un movimiento (tiempo limitado)
  - ❑ Ordenación de movimientos en función de los resultados de la última iteración
  - ❑ Uso de los resultados alfa y beta de la última iteración para inicializar los valores en la siguiente
    - ❑ Ayuda a efectuar más podas de forma temprana
- ❑ Efecto horizonte
  - ❑ Con límite fijo en la exploración, es más fácil que se produzca
  - ❑ Búsqueda de la quietud
    - ❑ Se continúa la búsqueda en esos nodos hoja
    - ❑ Deep Blue los explora hasta 30 capas por delante
- ❑ Paralelización *(resultó complicadísimo; > innovación de Deep Blue)*

Juegos: estado actual

- Programas que superan a los mejores jugadores humanos
  - Damas: CHINOOK (1994)
    - Derrotó al campeón mundial humano durante 40 años Marion Tinsley
    - BD de finales de partida: juego perfecto para configuraciones de tablero con 8 o menos piezas
  - Othello: LOGISTELLO (1997)
    - Los campeones humanos se niegan a medirse con programas tan buenos
  - Scrabble: MAVEN (1998)
- Programas competitivos con los mejores jugadores humanos
  - Ajedrez: DEEP BLUE (1997)
  - Backgammon: TD-GAMMON (1995, *aprendizaje por refuerzo*)
    - Complicación: aleatoriedad (tiradas de dados)
    - Los programas desarrollados por humanos son muy malos
    - En cambio, un sistema de aprendizaje máquina (mucha búsqueda y uso de los resultados para construir una muy buena función heurística) lo consiguió

Juegos: estado actual

- Juegos que presentan dificultades con los métodos actuales
  - Bridge
    - Información oculta (las cartas de los otros jugadores)
    - Comunicación con el compañero mediante un lenguaje restringido
    - Los jugadores máquina no son nada buenos en la fase del juego que involucra comunicación humana
  - Go
    - Como el ajedrez: información perfecta, no aleatoriedad, ni comunicación
    - Problema: el enorme factor de ramificación ($r > 300$)
      - Los métodos de búsqueda que funcionan bien en ajedrez no son susceptibles de aplicarse en el go
      - Los jugadores humanos parecen basarse en algo mucho más complejo: comprensión de patrones espaciales
      - Planteamiento de métodos basados en mejores heurísticas y menos en búsqueda por fuerza bruta, con bases de conocimiento de patrones
  - Poker

Observaciones

- Sobre juegos aplicables a la aproximación simbólica a la IA
  - Las máquinas superan a los humanos en áreas perfectamente definidas en las que las reglas están claras
    - Ajedrez
    - Matemáticas
  - Las áreas que siguen resultando extremadamente complicadas en la IA son más nebulosas
    - Lenguaje, visión y sentido común
    - La mayoría de la investigación actual está centrada en estas actividades no tan bien definidas
  - Los éxitos se han obtenido tras muchos años de refinamientos graduales incluso en actividades bien definidas como el ajedrez
    - No debemos esperar éxitos en el corto plazo en los grandes retos de la IA

Bibliografía

- **Russell, S. y Norvig, P.**
  Inteligencia Artificial: Un Enfoque Moderno.
  Prentice Hall, 2004, 2ª edición.
  - **Capítulos 3, 4 y 6**
- **Luger, G.F.**
  Artificial Intelligence.
  Addison-Wesley, 2005, 5ª edición.
  - **Capítulos 2, 3, 4 y 6**
- **Rich, E. y Knight, K.**
  Artificial Intelligence.
  McGraw-Hill, 1991, 2ª edición.
  - **Capítulos 2 y 3**

Bibliografía

- **Nilsson, J.**
  Artificial Intelligence: A New Synthesis.
  Prentice Hall, 2004, 2ª edición.
  - **Capítulos 7, 8, 9, 10, 11 y 12**
- **J. J. Rubio García, P. R. Muro Medrano y J. A. Bañares Bañares**
  Apuntes de búsqueda para IAIC 1.
  Dpto. de Informática e Ingeniería de Sistemas, Universidad de Zaragoza, 1998, versión 1.0.
- **T. Lozano-Perez y L. Kaelbling**
  Artificial Intelligence.
  Electrical Engineering and Computer Science, MIT OpenCourseWare, Massachusetts Institute of Technology, 2003.
  - **Capítulo 2: *Search Handout***