



UNIVERSIDAD
DE GRANADA



Inteligencia Artificial

Tema 4: Búsqueda con adversario: juegos

E.T.S. de Ingenierías Informática y de Telecomunicación
Universidad de Granada

© Antonio González

gonzalez@ugr.es

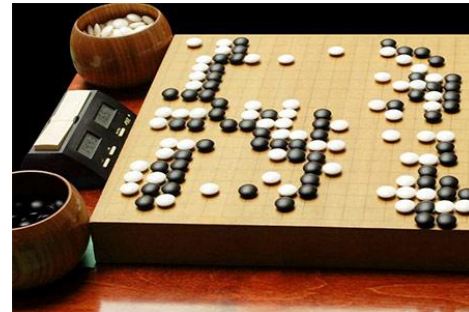
**Departamento de Ciencias de la
Computación e Inteligencia Artificial**
<http://decsai.ugr.es>

*Este documento está protegido por la Ley
de Propiedad Intelectual ([Real Decreto Ley
1/1996 de 12 de abril](#)).*

*Queda expresamente prohibido su uso o
distribución sin autorización del autor.*

Objetivos

- Conocer las técnicas básicas de búsqueda con adversario (minimax, poda alfa-beta) y su relación con los juegos.



Estudia el tema en ...

- Nils J. Nilsson, “*Inteligencia Artificial: Una nueva síntesis*”, Ed. Mc Graw Hill, 2000. pp. 175-192
- S. Russell, P. Norvig, Artificial Intelligence: A modern Approach, Tercera Edición, Ed. Pearson, 2010.
- S. Russell, P. Norvig, Artificial Intelligence: A modern Approach, Cuarta Edición, Ed. Pearson, 2022.

Contenido

- Juegos bipersonales con información perfecta
- Árboles de exploración de juegos
- El modelo básico
- Juegos en los que interviene un elemento aleatorio

Interés

- Laboratorios perfectos para investigar en técnicas de resolución de problemas.
- Es fácil medir el éxito o el fracaso.
- Fascinación para cierta gente.
- Aspecto comercial.
- Aplicaciones en ámbitos empresariales.

Juegos bipersonales con información perfecta

- Estas situaciones se estudian y resuelven utilizando la **Teoría de Juegos**. La teoría matemática de juegos fue inventada como tal por **John von Neumann** y por **Oskar Morgenstern** en 1944.
- **¿Qué es un juego?**
 - Es cualquier situación de decisión, caracterizada por poseer una interdependencia estratégica, gobernada por un conjunto de reglas y con un resultado bien definido.
- En un juego, cada jugador intenta conseguir el mayor beneficio para sus intereses. La solución de un juego permite indicar a cada jugador qué resultado puede esperar y cómo alcanzarlo.

Juegos bipersonales con información perfecta

- **Ejemplo de juego: El dilema del prisionero**

- Dos individuos son detenidos por la policía debido a que cometieron cierto delito. Ambos son encerrados en celdas diferentes y son interrogados de forma individual. Ambos tienen dos alternativas: no confesar o delatar al compañero. Saben que si ninguno confiesa, ambos irán a la cárcel por 2 años, pero si uno delata a su compañero y el otro no, entonces al que confiesa le absuelven y al otro le encierran por 10 años. Si ambos confesasen, entonces la pena se repartiría y ambos irían a prisión por 5 años.

Juegos bipersonales con información perfecta

- **Ejemplo de juego: El dilema del prisionero**

		Prisionero 1	
		No delatar	Delatar
Prisionero 2	No delatar	$(-2, -2)$	$(0, -10)$
	Delatar	$(-10, 0)$	$(-5, -5)$

- **¿Qué harán los prisioneros?** Con toda lógica: Cooperar. Sin embargo, la tentación de hacer la promesa de no delatar, para después traicionar al compañero es muy grande.
- El juego tiene una estructura no cooperativa.

Juegos bipersonales con información perfecta

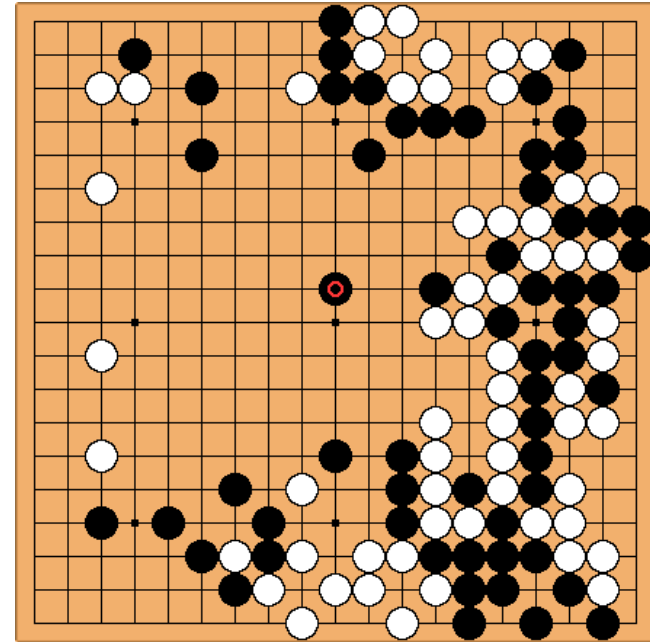
- **Ejemplo de juego:** El juego de los palillos

- Inicialmente, hay n palillos sobre la mesa, y dos jugadores A y B. El jugador A comienza el juego quitando 1, 2 ó 3 palillos. Le sigue el jugador B, que también podrá quitar 1, 2 ó 3 palillos. El turno vuelve al jugador A, y estas acciones se repiten hasta que quede un único palillo en la mesa. Aquel que quite este último palillo pierde el juego.



- **Pregunta:** ¿Cómo debe jugar A para maximizar su beneficio?

Juegos bipersonales con información perfecta

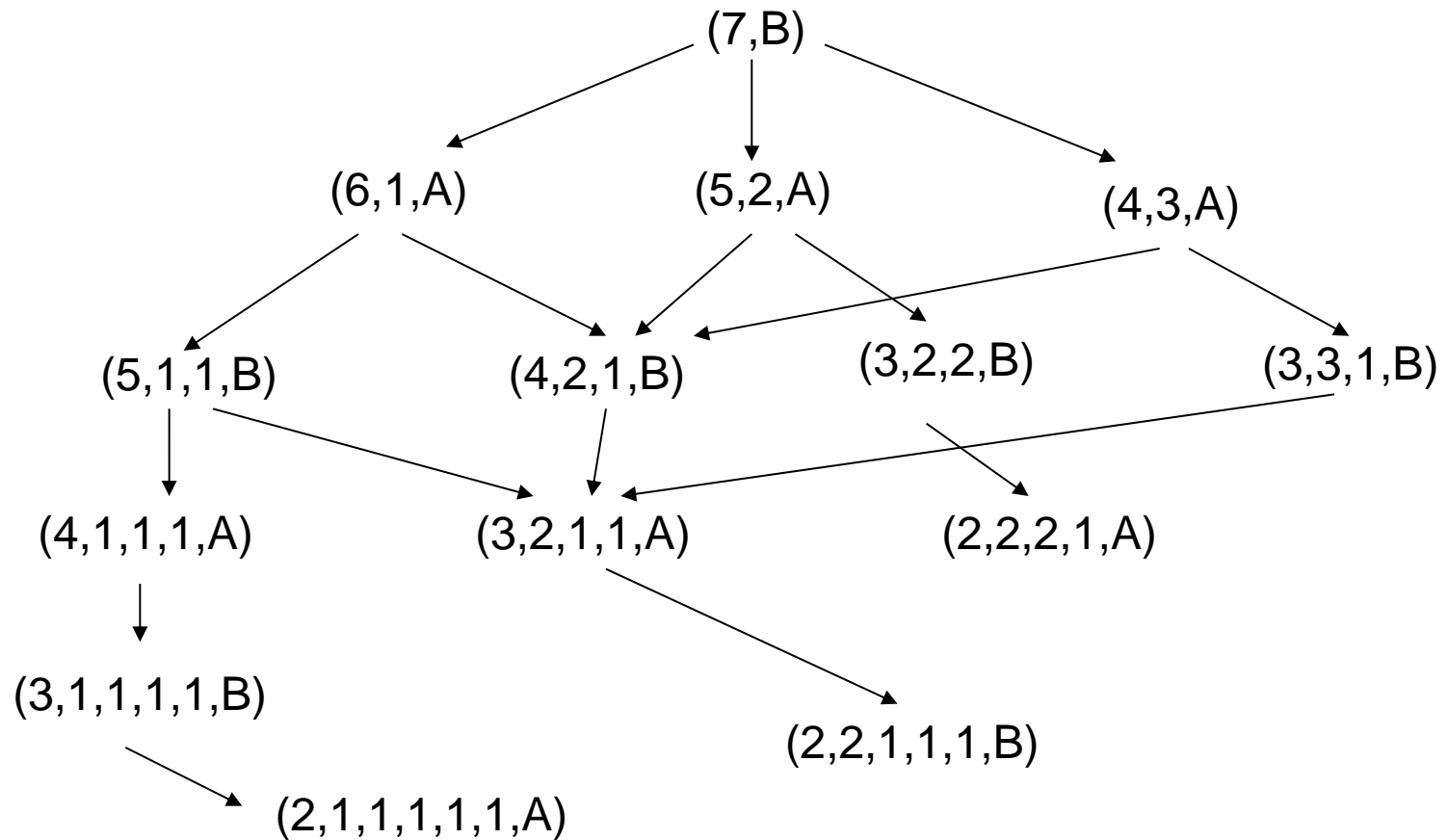


- Un juego de **información perfecta** es aquel en los jugadores tienen a su disposición toda la información de la situación del juego.

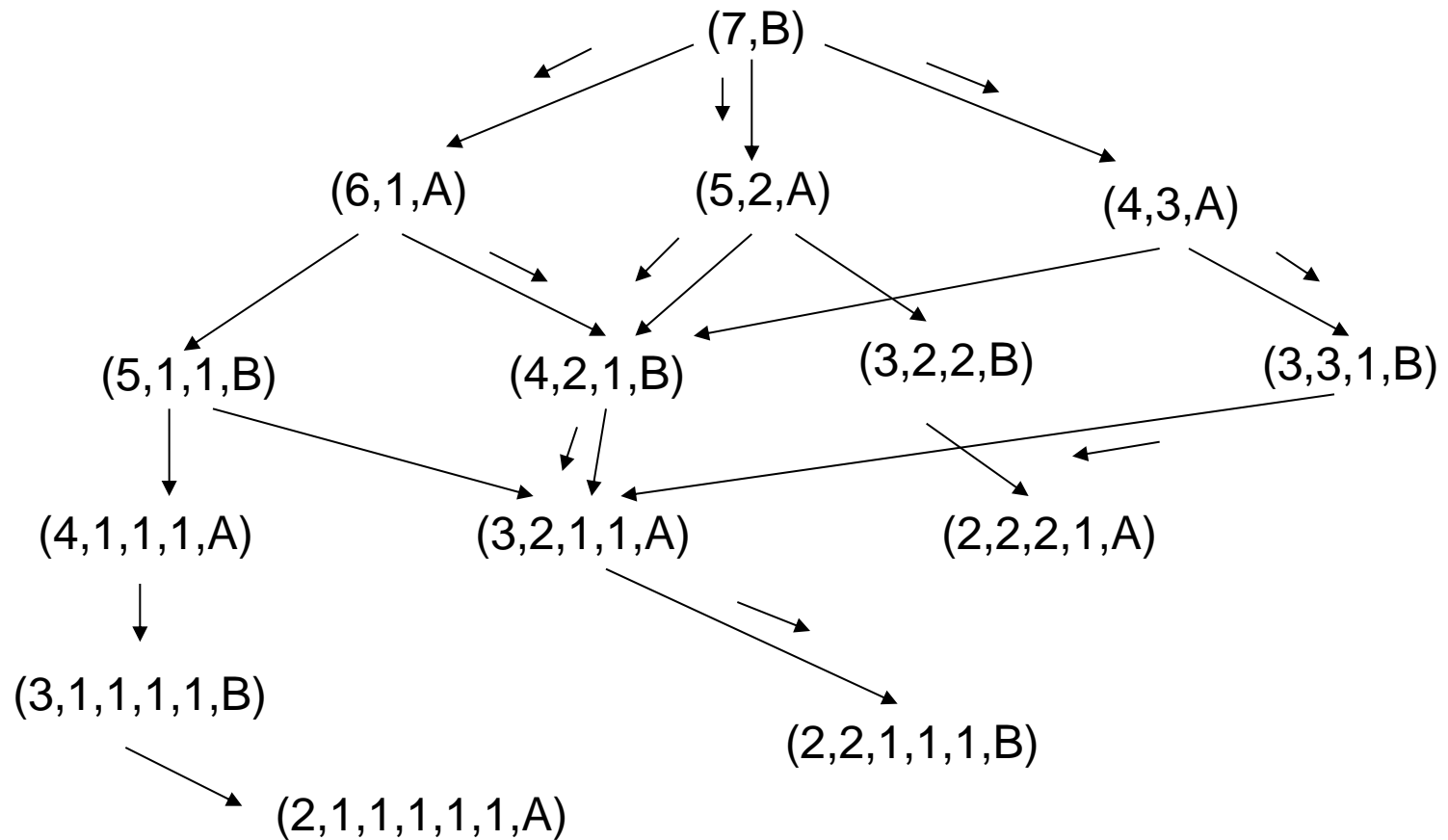
Árboles de exploración de juegos

- Un árbol del juego es una representación explícita de todas las formas de jugar a un juego
- Correspondencia entre árboles de juegos y árboles Y/O

Ejemplo simple



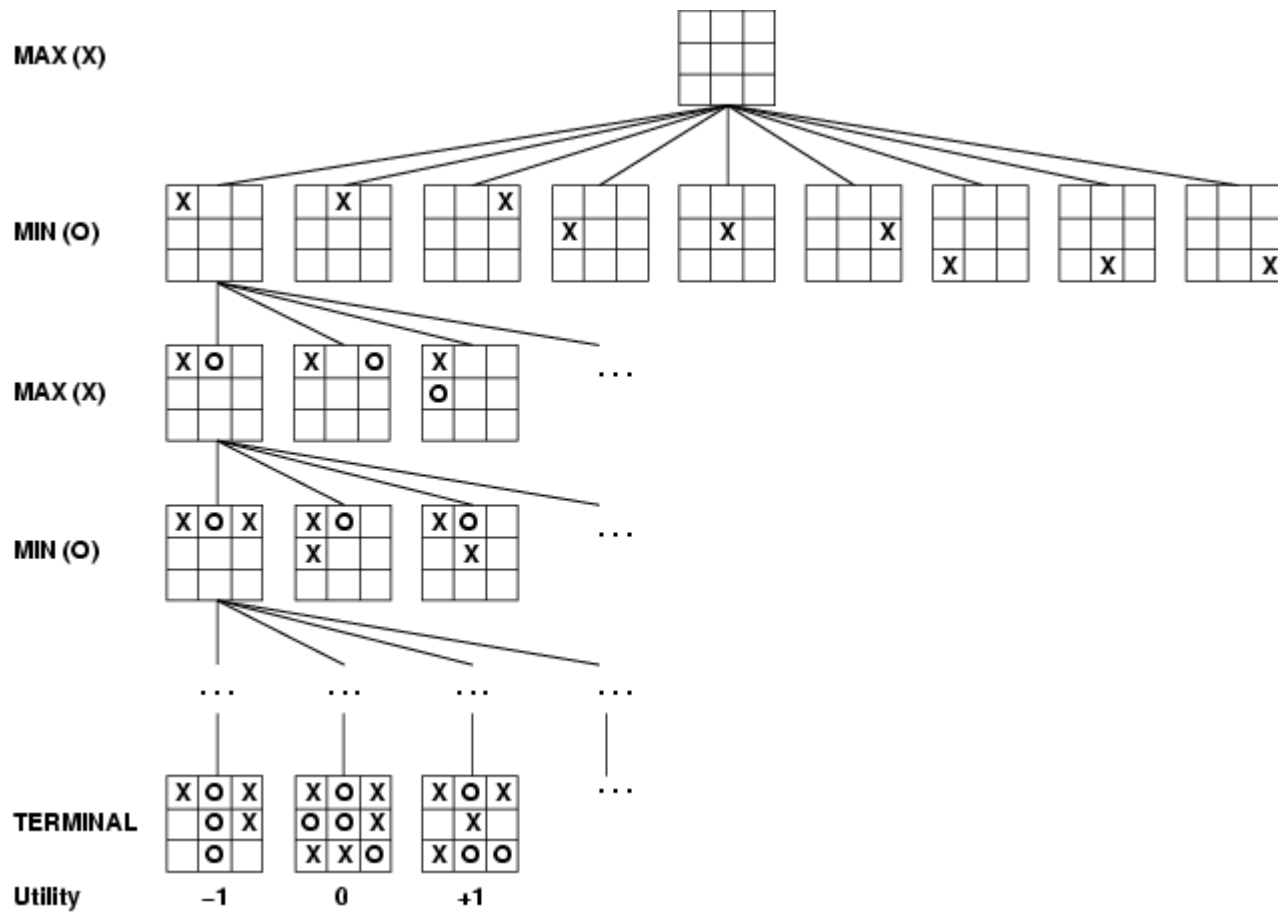
Resolución del ejemplo



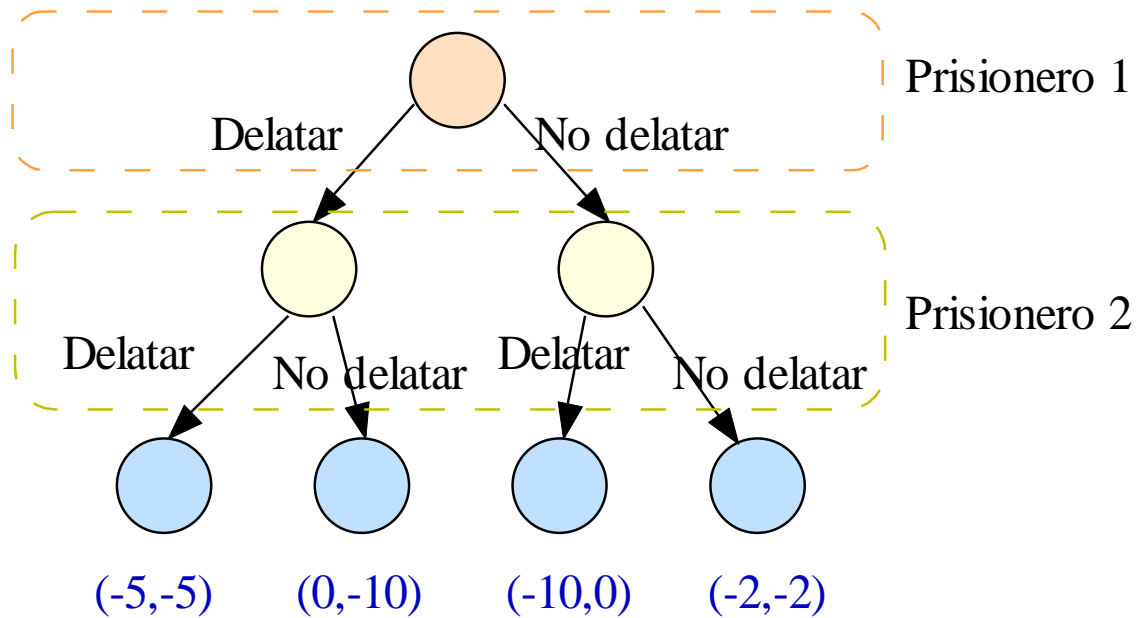
Notación min-max

- MAX: primer jugador
- MIN: segundo jugador
- Nodos MAX y nodos MIN
- Los nodos terminales se etiquetan con V, D o E desde el punto de vista de MAX

Ejemplo



Ejemplo



Algoritmo STATUS

- Si J es un nodo MAX no terminal, entonces STATUS(J)=
 - V si alguno de los sucesores de J tiene STATUS V
 - D si todos los sucesores de J tienen STATUS D
 - E en otro caso
- Si J es un nodo MIN no terminal, entonces STATUS(J)=
 - V si todos los sucesores de J tienen STATUS V
 - D si alguno de los sucesores de J tiene STATUS D
 - E en otro caso

Nuevo modelo de solución

- Los juegos complejos no se pueden resolver ya que es imposible la exploración total hasta la terminación
- Nuevo objetivo: encontrar una buena jugada inmediata
- Importancia de la heurística en el proceso

El modelo básico

- Arquitectura
percepción/planificación/actuación
- Búsqueda con horizonte
- Uso de heurísticas

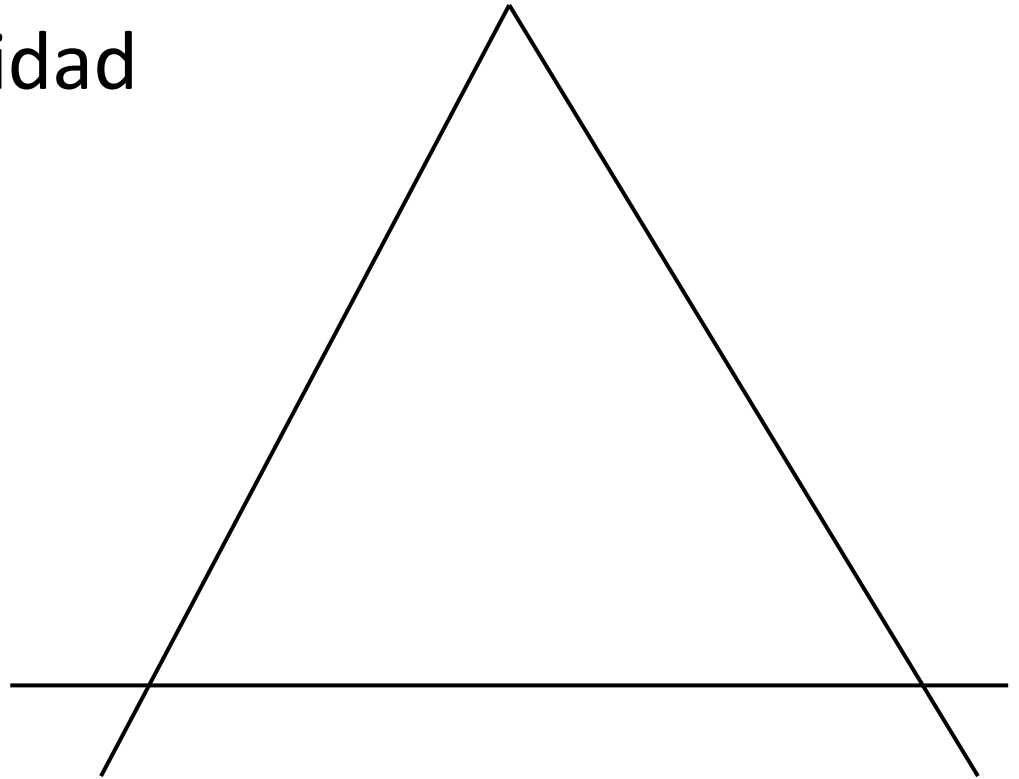
El modelo básico

- Horizonte: profundidad
- Heurística
- Búsqueda parcial
- Propagación

Complejidad de un juego:

B^P

Con B factor de ramificación y P la profundidad



Heurísticas

- Heurística para el ajedrez del programa de Turing: B/N
- Heurística para las damas del programa de Samuel: función lineal de varias características

$$f(x) = \alpha_1 * x_1 + \alpha_2 * x_2 + \dots + \alpha_n * x_n$$

La regla minimax

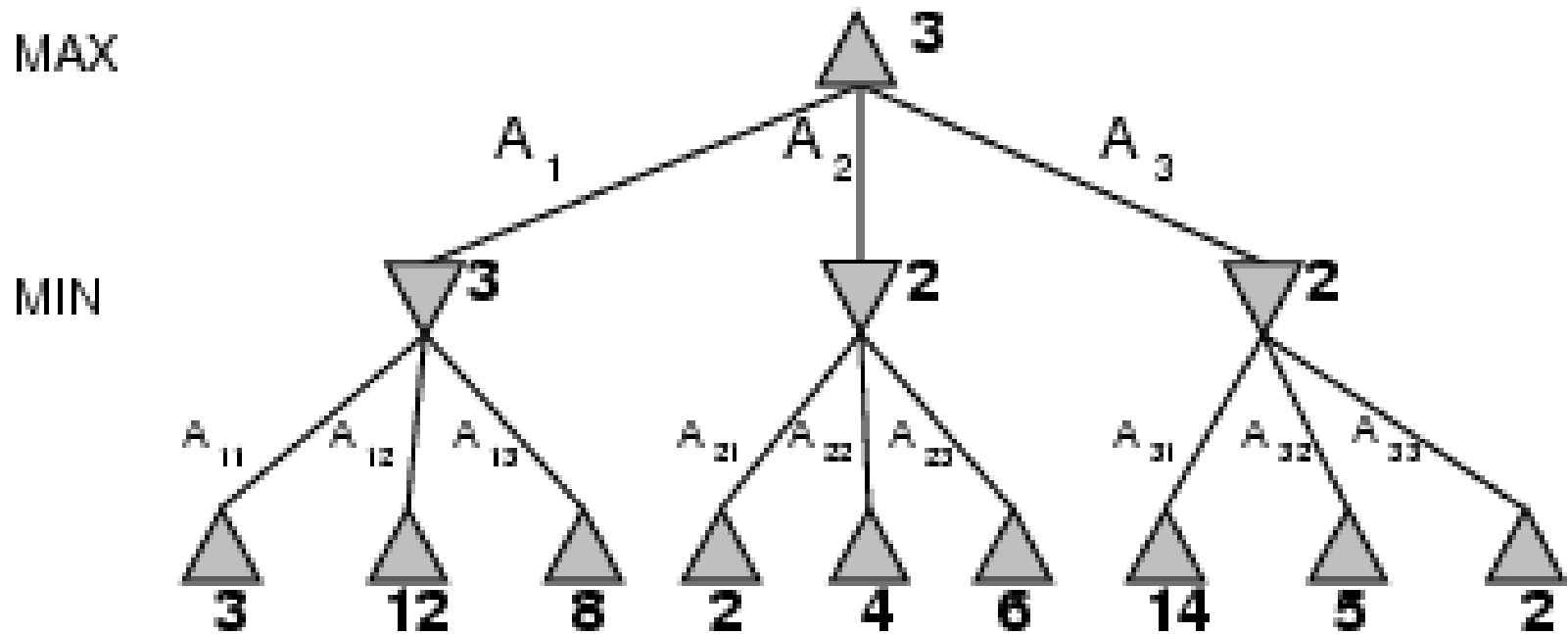
- El valor $V(J)$ de un nodo J de la frontera de búsqueda es igual al de su evaluación estática; en otro caso
- Si J es un nodo MAX, entonces su valor $V(J)$ es igual al máximo de los valores de sus nodos sucesores
- Si J es un nodo MIN, entonces su valor $V(J)$ es igual al mínimo de los valores de sus nodos sucesores.

Algoritmo Minimax

Para determinar el valor minimax, $V(J)$ de un nodo J , hacer lo siguiente:

- Si J es un nodo terminal, devolver $V(J)=f(J)$; en otro caso
- Para $k=1,2,\dots,b$, hacer:
 - Generar J_k , el k -ésimo sucesor de J
 - Calcular $V(J_k)$
 - Si $k=1$, hacer $AV(J) \leftarrow V(J_1)$; en otro caso, para $k \geq 2$,
 - hacer $AV(J) \leftarrow \max\{AV(J), V(J_k)\}$ si J es un nodo MAX o
 - hacer $AV(J) \leftarrow \min\{AV(J), V(J_k)\}$ si J es un nodo MIN
- Devolver $V(J)=AV(J)$

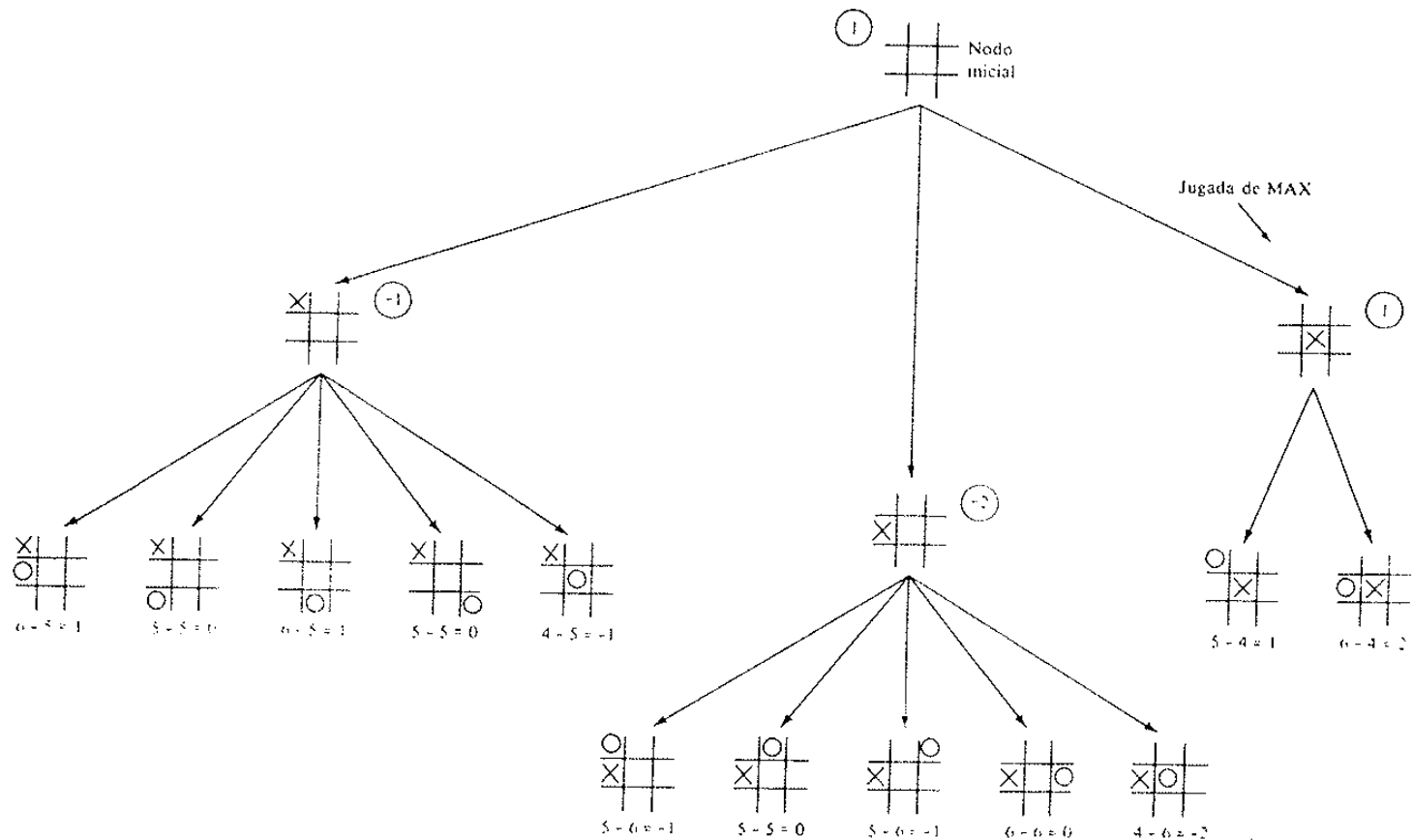
Ejemplo



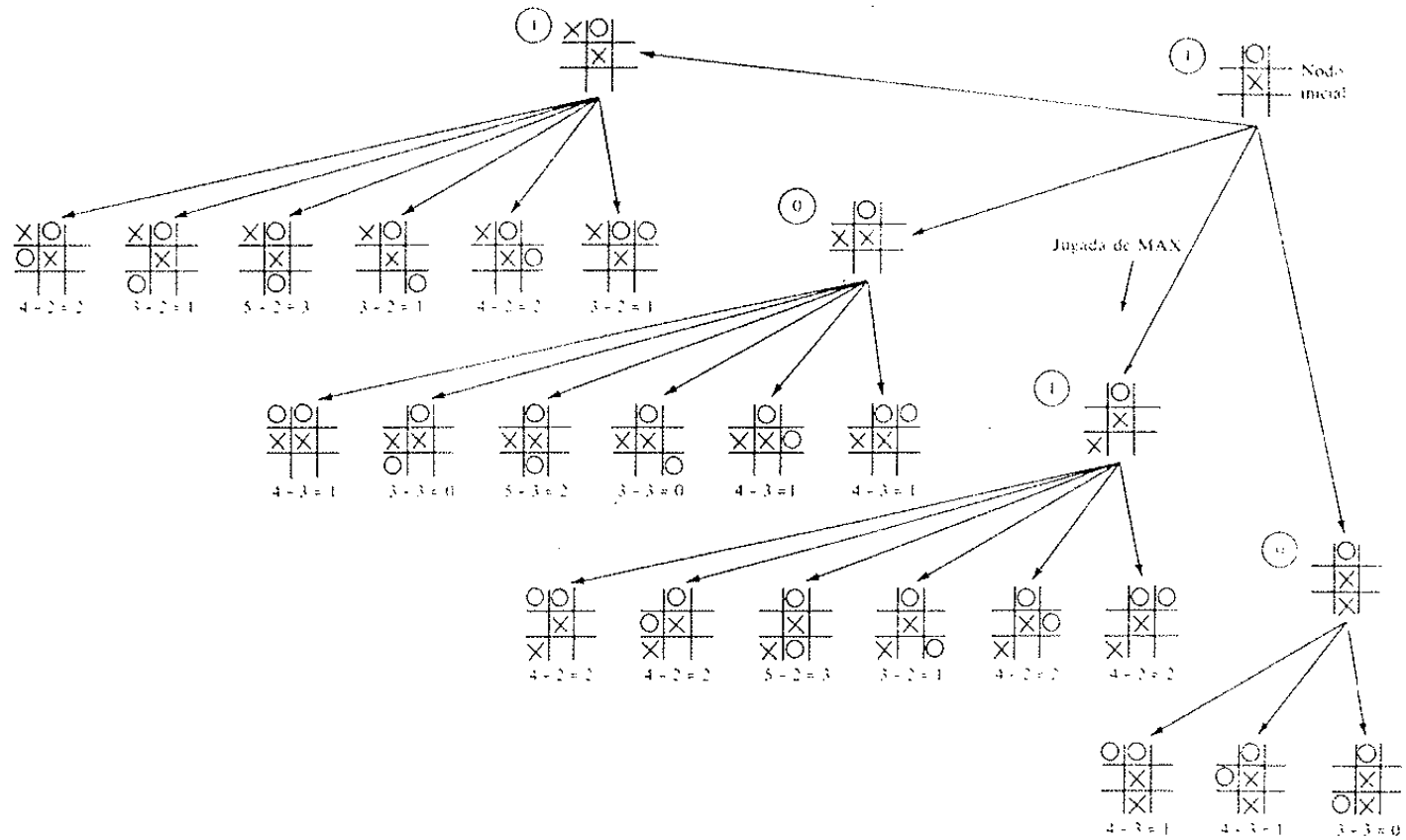
Ejemplo

- Juego del tres en raya
- Profundidad 2
- Heurística
 - Lo que es bueno para max – lo que es bueno para min
 - Contemplar casos extremos

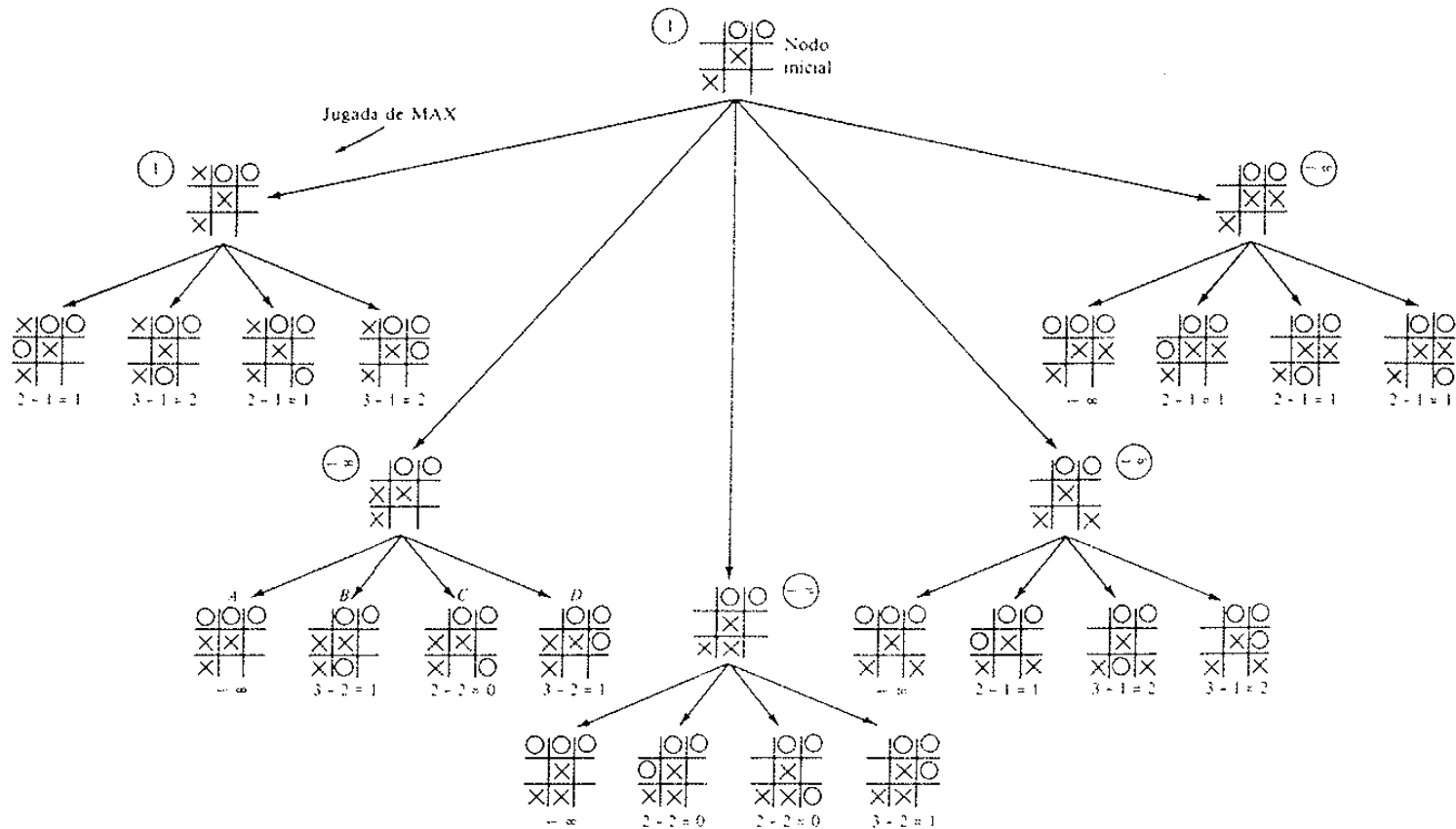
Ejemplo



Ejemplo



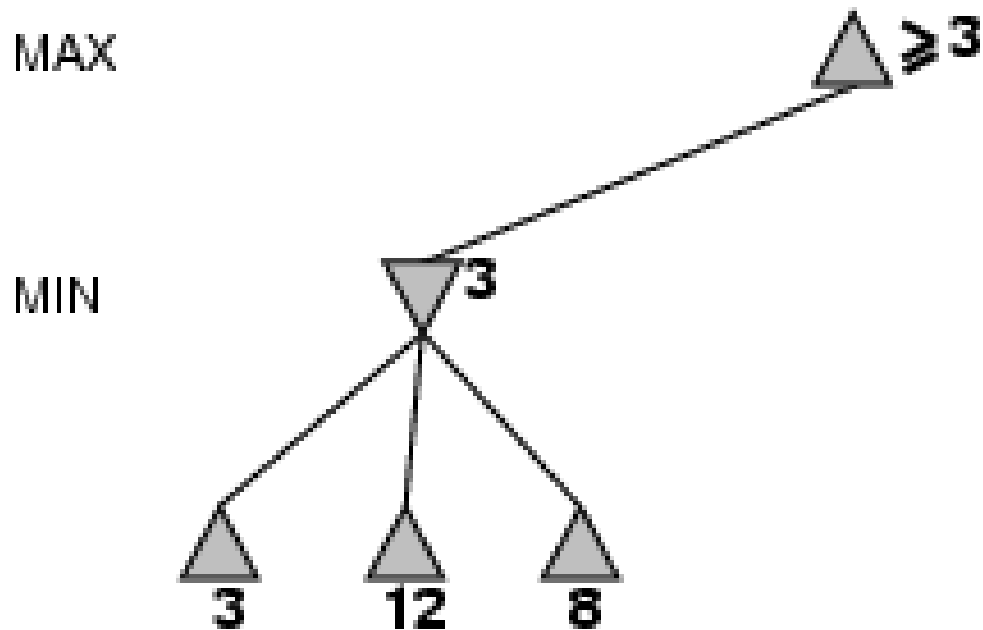
Ejemplo



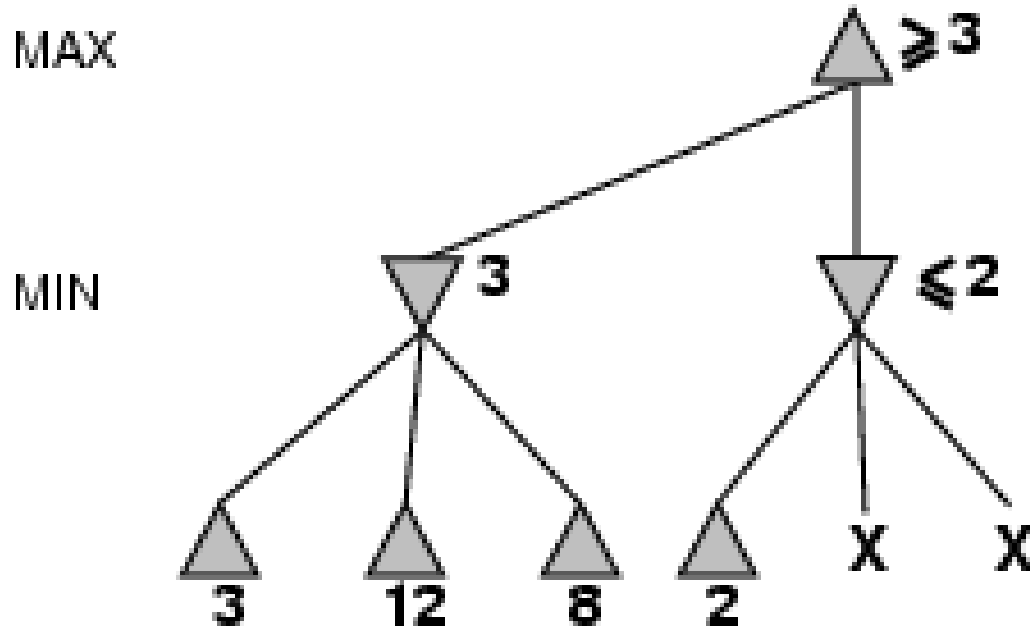
Poda alfa-beta

- ¿podríamos obtener el mismo resultado que el algoritmo minimax con menos esfuerzo computacional?

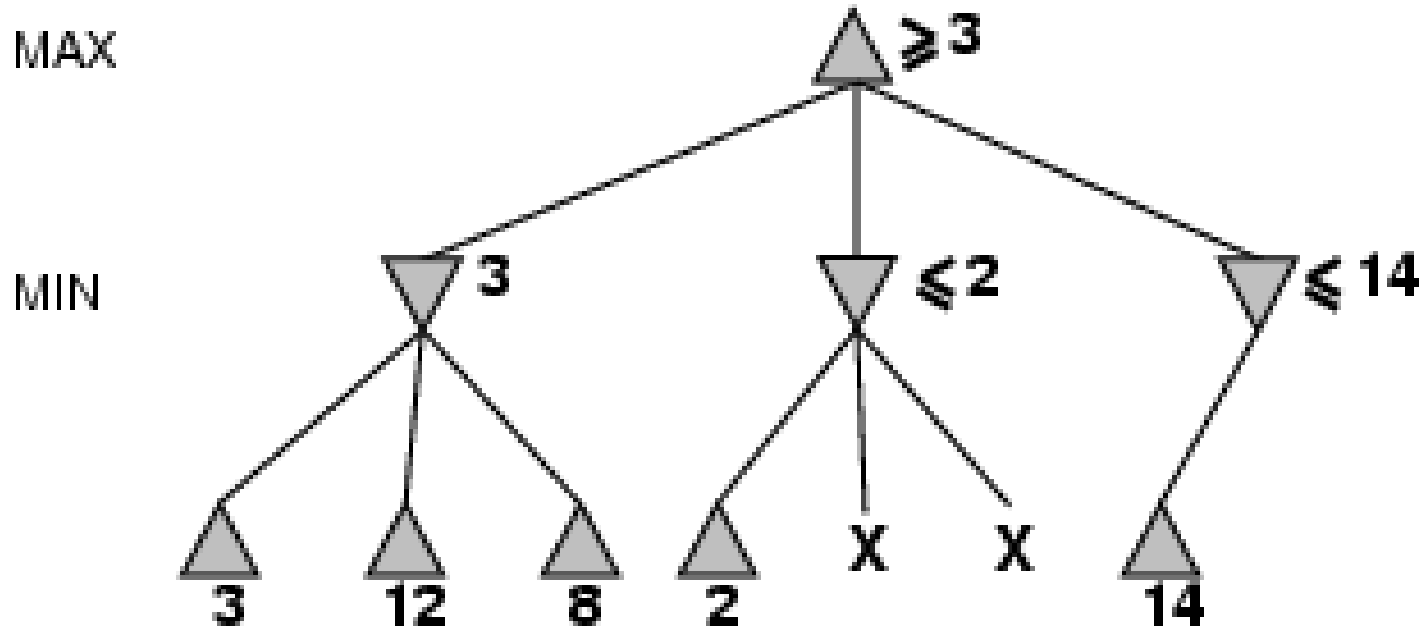
Ejemplo poda



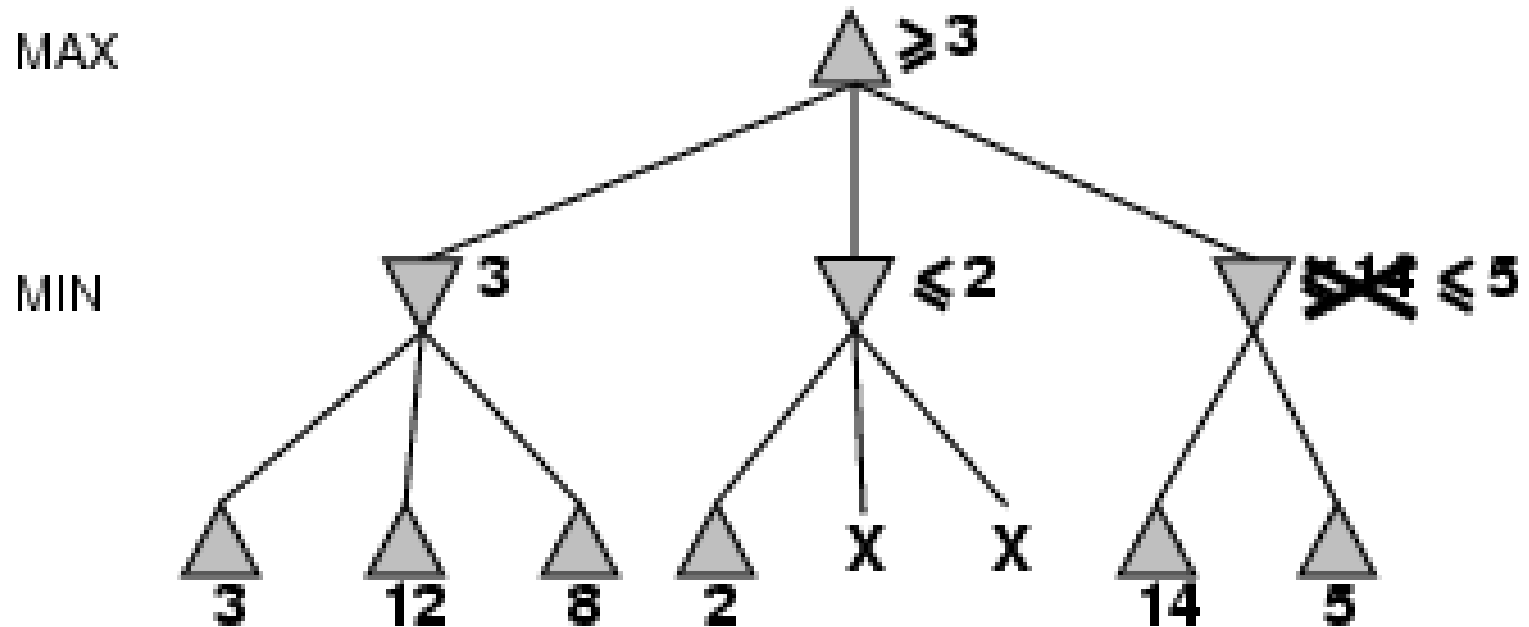
Ejemplo poda



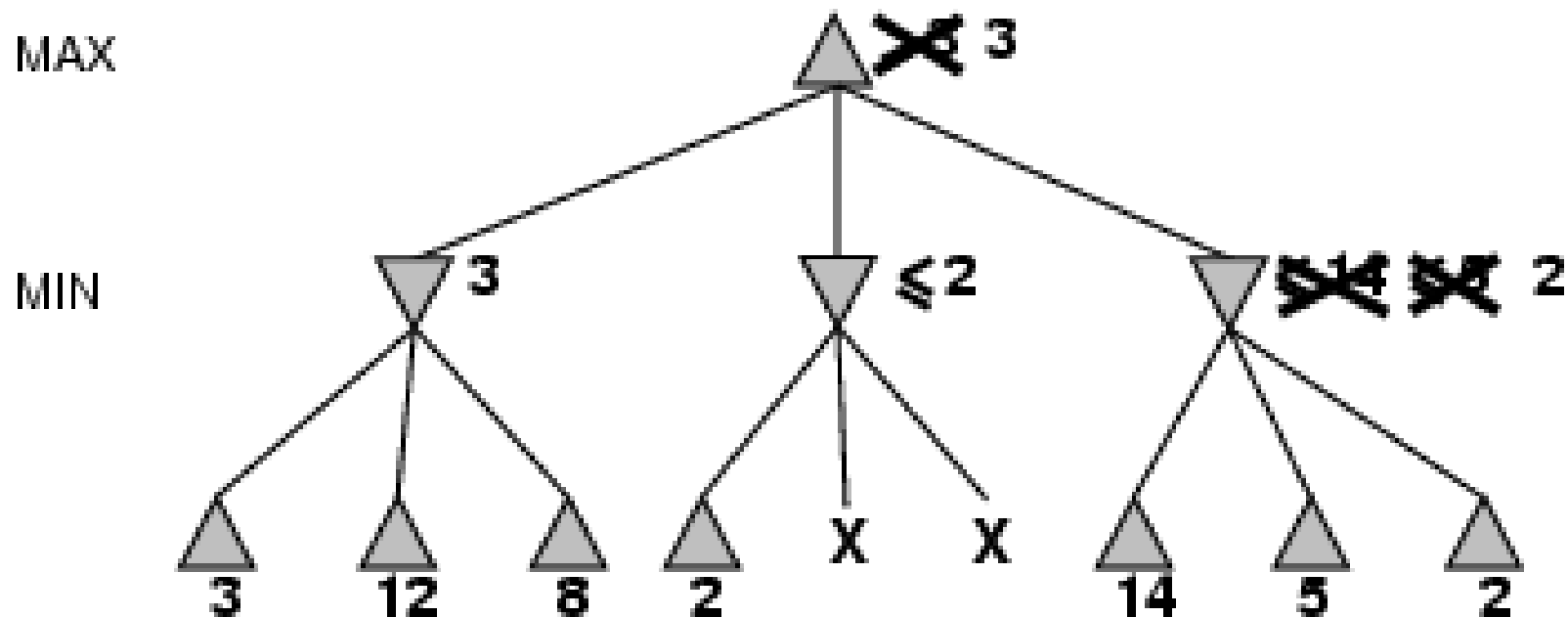
Ejemplo poda



Ejemplo poda



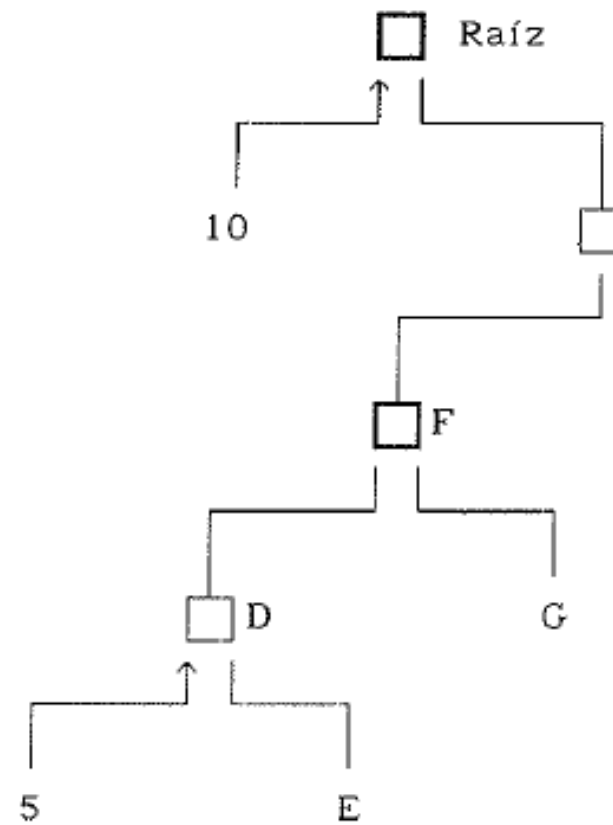
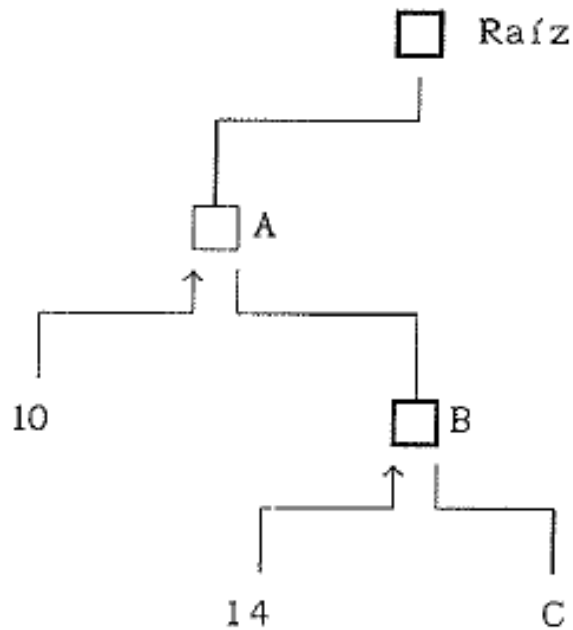
Ejemplo poda



Cotas alfa y beta

	Para nodos	Se calcula	es
Cota alfa	Nodos MIN	Máximo de los nodos MAX	Cota inferior
Cota beta	Nodos MAX	Mínimo de los nodos MIN	Cota superior

Cotas alfa y beta



Algoritmo ALFA-BETA

Para calcular el valor $V(J, \alpha, \beta)$, hacer lo siguiente:

1. Si J es un nodo terminal, devolver $V(J)=f(J)$. En otro caso, sean $J_1, \dots, J_k, \dots, J_b$ los sucesores de J . Hacer $k \leftarrow 1$ y, si J es un nodo MAX ir al paso 2; si J es un nodo MIN ir al paso 5.
2. Hacer $\alpha \leftarrow \max(\alpha, V(J_k, \alpha, \beta))$.
3. Si $\alpha \geq \beta$ devolver β ; si no, continuar
4. Si $k=b$, devolver α ; si no, hacer $k \leftarrow k+1$ y volver al paso 2.
5. Hacer $\beta \leftarrow \min(\beta, V(J_k, \alpha, \beta))$.
6. Si $\beta \leq \alpha$ devolver α ; si no, continuar
7. Si $k=b$, devolver β ; si no, hacer $k \leftarrow k+1$ y volver al paso 5.

Complejidad

En el peor caso: B^P

En el mejor caso:

$$2B^{P/2} - 1, \text{ si } P \text{ es par,}$$

$$B^{(P+1)/2} + B^{(P-1)/2} - 1, \text{ si } P \text{ es impar,}$$

En el caso promedio: nos permite profundizar un 33%

Heurísticas para la búsqueda en árboles de juegos

- Programa tipo A
- Programa tipo B
 - Punto de parada razonable
 - Selección sobre las líneas del juego

Heurísticas para la búsqueda en árboles de juegos

- Evaluación hacia atrás
- Profundidad de la búsqueda
- Ordenación de la búsqueda
- Anchura de la búsqueda
- Alternativas de la búsqueda

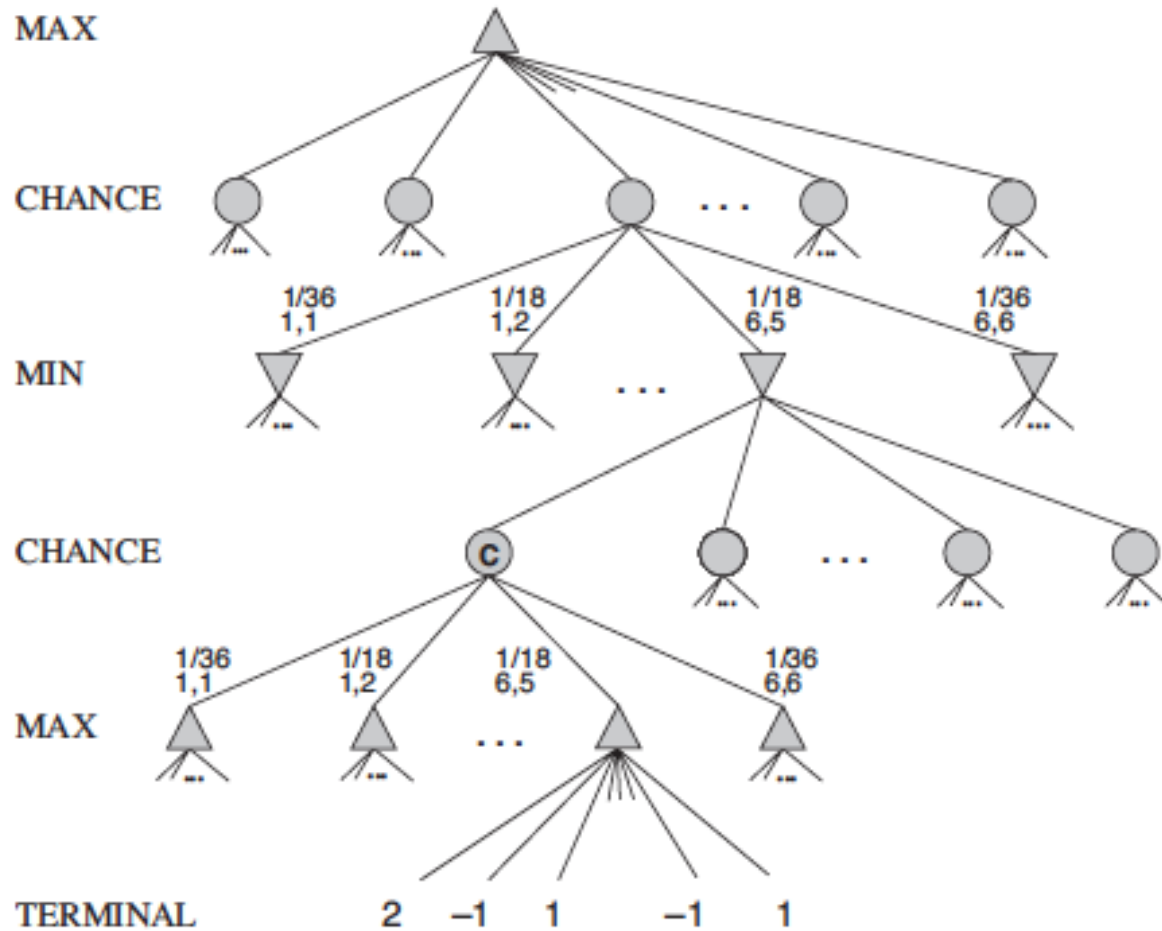
Los mejores motores de ajedrez

- [Leela Chess Zero](#) y [Stockfish](#) son, a día de hoy, los mejores motores de ajedrez, exponentes de la excelencia en el juego lograda con esta primera aproximación.

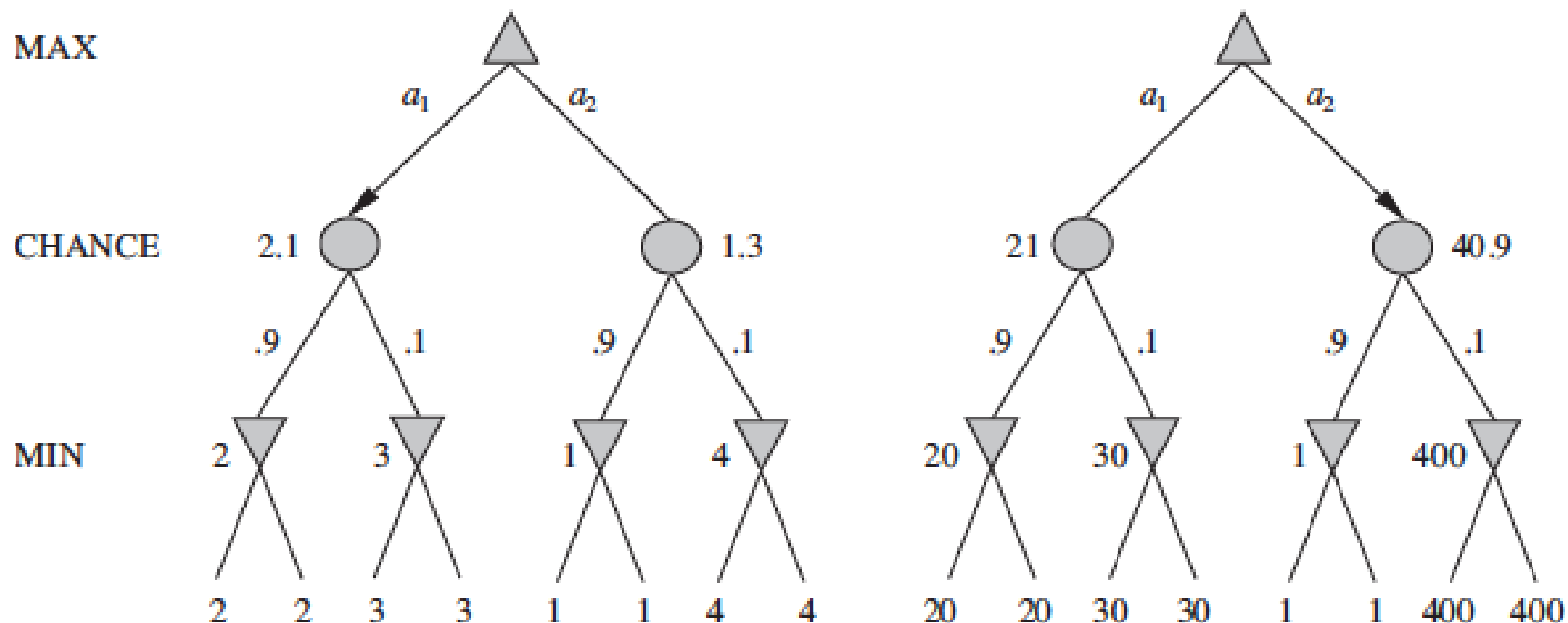
Juegos en los que interviene un elemento aleatorio



Modelo



Algunos problemas



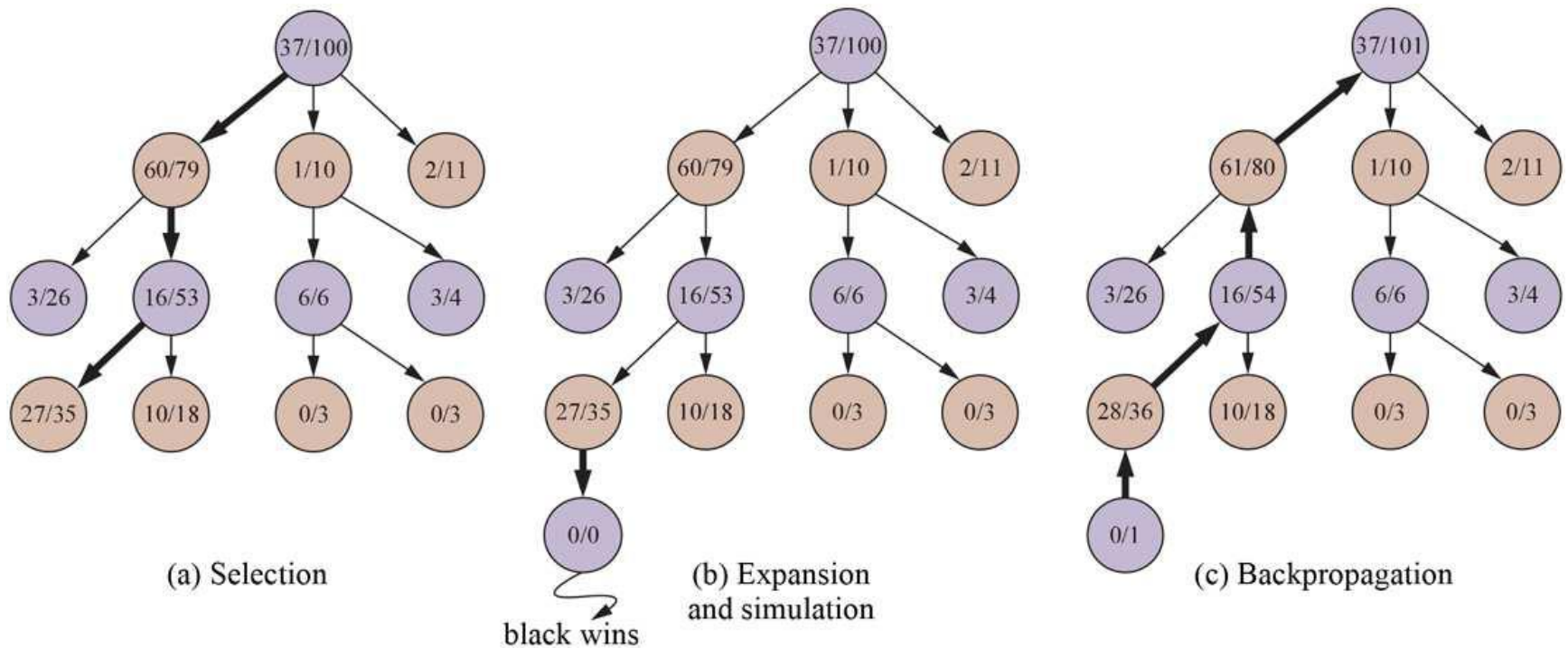
Búsqueda en Árboles Monte Carlo

- Las principales debilidades de la poda alfa-beta son:
 - La complejidad cuando el factor de ramificación crece.
 - La definición de una buena función heurística
- Evolución de los juegos hacia una estrategia llamada Búsqueda en Árboles Monte Carlo (MCTS Monte Carlo Tree Search).

Idea de MCTS

- El modelo básico de MCTS no hace uso de funciones heurísticas.
- Estima el valor de un estado como la utilidad promedio sobre un número de simulaciones de juegos completos empezando en dicho estado.
- Hay que determinar que jugada hace cada jugador durante la simulación:
 - Aleatoria.
 - Mediante políticas.
- Exploración/Explotación.

Idea del MCTS



AlphaZero

- [AlphaZero](#), el programa desarrollado por DeepMind, la división de inteligencia artificial de Google, y sobre el cual la revista Science ha publicado recientemente un artículo.
- AlphaZero es capaz de vencer a los juegos de mesa más complejos para la mente humana: el ajedrez, el shogi (ajedrez japonés) y el go (juego de tablero tradicional chino).

ChessBench

- ChessBench
 - Para su entrenamiento y aprendizaje se han utilizado 10 millones de partidas de ajedrez
 - A partir de esas partidas se generan más de 15000 millones de anotaciones evaluando cada posición y su mejor acción (movimiento). Esta evaluación la lleva a cabo Stockfish 16, cuyo motor de búsqueda determina el mejor movimiento posible en una posición concreta.