

# Tema 4: Búsqueda con adversario: juegos



# 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.

# Contenido

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

# Juegos

- Hasta ahora hemos considerado un solo agente reactivo/deliberativo.
    - El espacio (árbol/grafó) de búsqueda se genera a partir de solo sus propias acciones
    - Por cada estado, él decide qué acción tomar, y en el nuevo estado resultante, él vuelve a controlar qué acción tomar.
    - ¿qué ocurre con más de un agente?
  - Entorno multiagente
    - Cualquier agente necesita considerar las acciones de otros agentes y cómo afectan a su propio estado.
    - Cooperativo: agentes trabajan para alcanzar un objetivo común
    - Competitivo: el objetivo de cada agente entra en conflicto con los del resto.
  - Búsqueda con adversario: problemas de búsqueda en **entornos multiagente competitivos** que a partir de ahora llamaremos **juegos**.
-

# 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.
  - Entorno multiagente visto como un juego en el que el impacto de cada agente sobre sus pares es significativo. A menudo en economía muchos agentes se ven como **economías** en lugar de **juegos**.
  - Usada para modelar decisiones en este tipo de entornos

# Juegos bipersonales con información perfecta

- ¿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	$(-10, 0)$	$(-5, -5)$

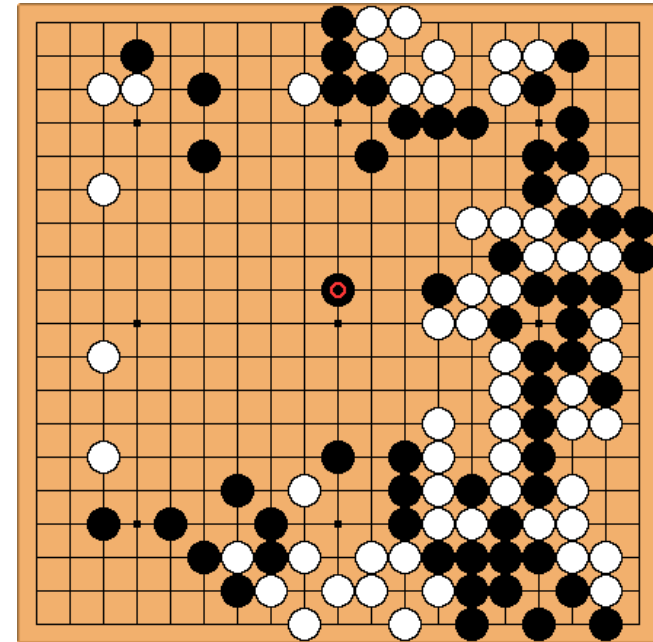
- **¿Qué harán los prisioneros?** Con toda lógica: Cooperar. Sin embargo, la tentación de hacer la promesa de no delatar, para después traicionar al compañero es muy grande.
- El juego tiene una estructura no cooperativa.

# 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?

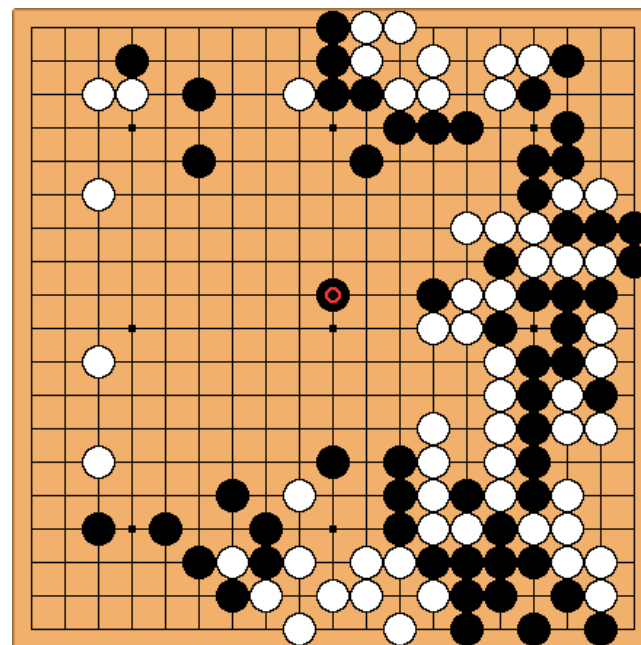


# Juegos bipersonales con información perfecta



- Juegos de **suma nula** (zero-sum games): en la situación final el beneficio de un jugador es total y la pérdida del oponente es total, o hay empate.
- Es decir las valoraciones de los estados finales del juego son o bien iguales , o bien opuestas.
- Por ejemplo, si un jugador gana al ajedrez (valoración +1), el otro necesariamente pierde (valoración -1).

# Juegos bipersonales con información perfecta



- Juegos de suma nula (zero-sum games): en la situación final el beneficio de un jugador es total y la pérdida del oponente es total, o hay empate
- Juegos de **suma no nula** (non zero-sum games) hay situaciones finales en las que se distribuyen las pérdidas y ganancias.
- Un juego de **información perfecta** es aquel en los jugadores tienen a su disposición toda la información de la situación del juego.

Nos centraremos en juegos determinísticos, bipersonales, por turnos, de suma nula y con información perfecta.

# Juegos como problema de búsqueda

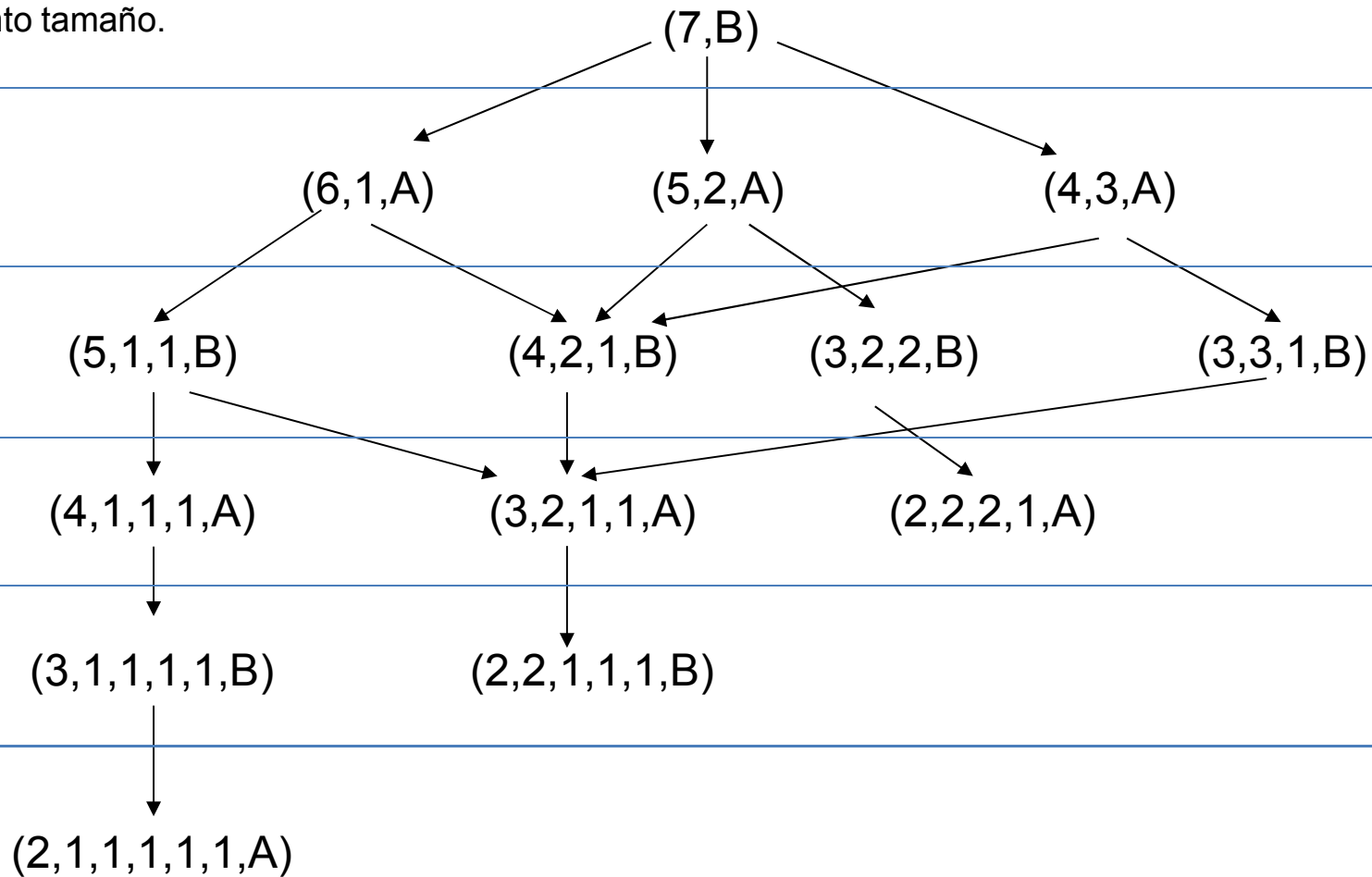
- **Estado inicial:** donde se representa la posición inicial del tablero y se identifica el jugador que mueve.
- **Función sucesor:** devuelve una lista de pares (*movimiento, estado*), cada una indicando un movimiento legal y el estado resultante.
- **Test terminal**, función que determina cuándo un juego ha finalizado. Los estados donde el juego finaliza se llaman **estados terminales**.
- **Función de valoración:** (función de utilidad) devuelve un valor numérico para estados terminales. Ajedrez (V,D,E o +1, -1, 0). En otros juegos hay variedad de posibles resultados (por ejemplo puntos ganados...).

# Árboles de exploración de juegos

- Un árbol del juego es una representación explícita de todas las formas de jugar a un juego
  - El estado inicial más todos los movimientos legales forman un **árbol de juego**.
- Correspondencia entre árboles de juegos y árboles Y/O

# Ejemplo simple

Movimiento: Reducir una pila de objetos en dos pilas con distinto tamaño.

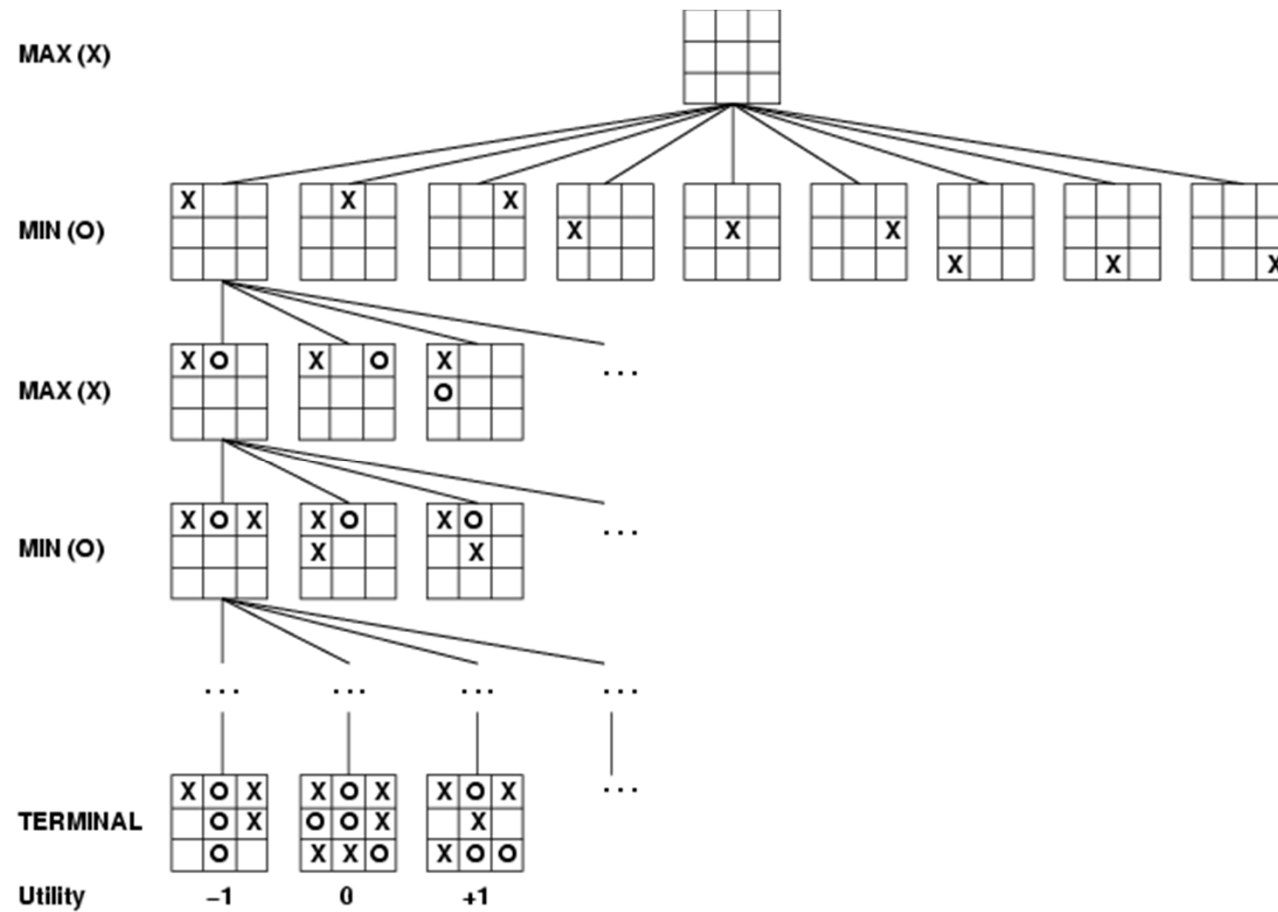




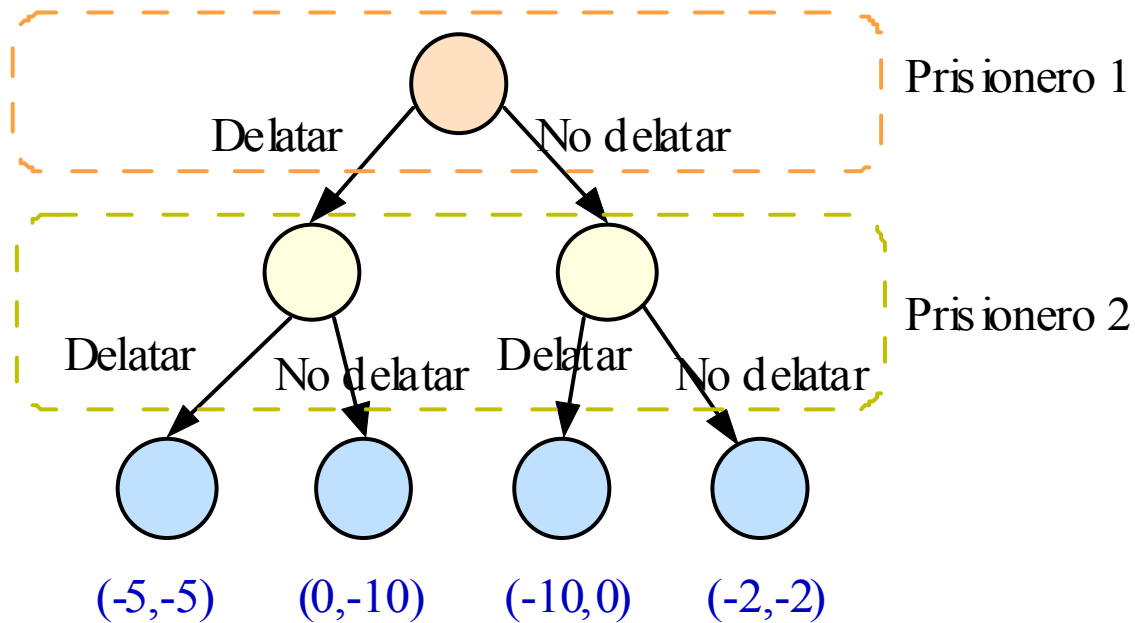
# Notación min-max

- 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



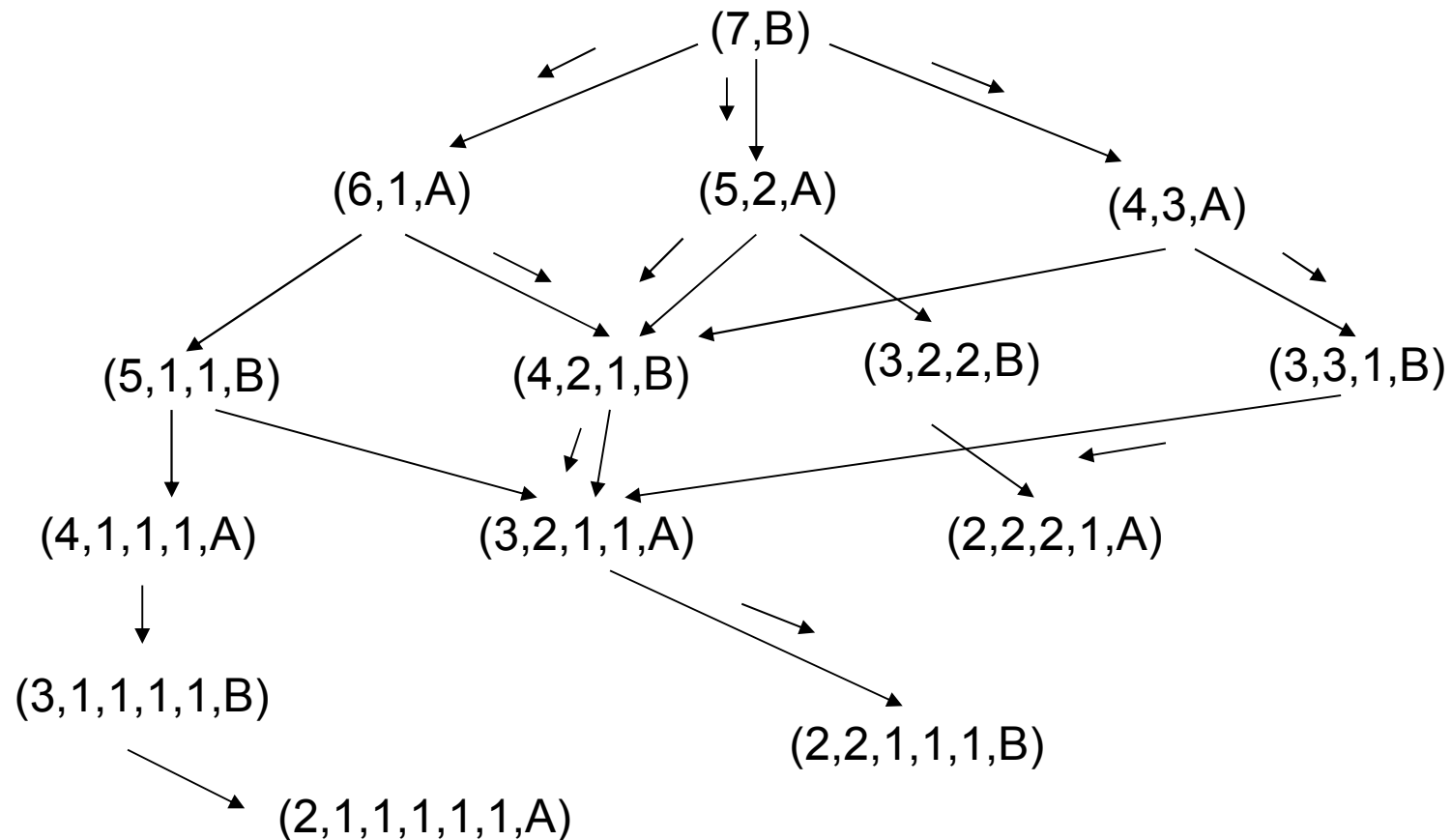
# Ejemplo



# Estrategia

- En un problema de búsqueda normal, la solución es una secuencia de movimientos que llevan a un estado objetivo (un estado terminal que es victoria).
- En un juego MIN tiene algo que decir al respecto.
- MAX debe encontrar una **estrategia** contingente (que tiene en cuenta los movimientos de MIN).

# Resolución del ejemplo



# Resolver un juego

- ¿Qué significa resolver un juego?
  - Encontrar un valoración para el nodo inicial.
  - Determinar una estrategia ganadora para MAX o para MIN.

# 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

# 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.

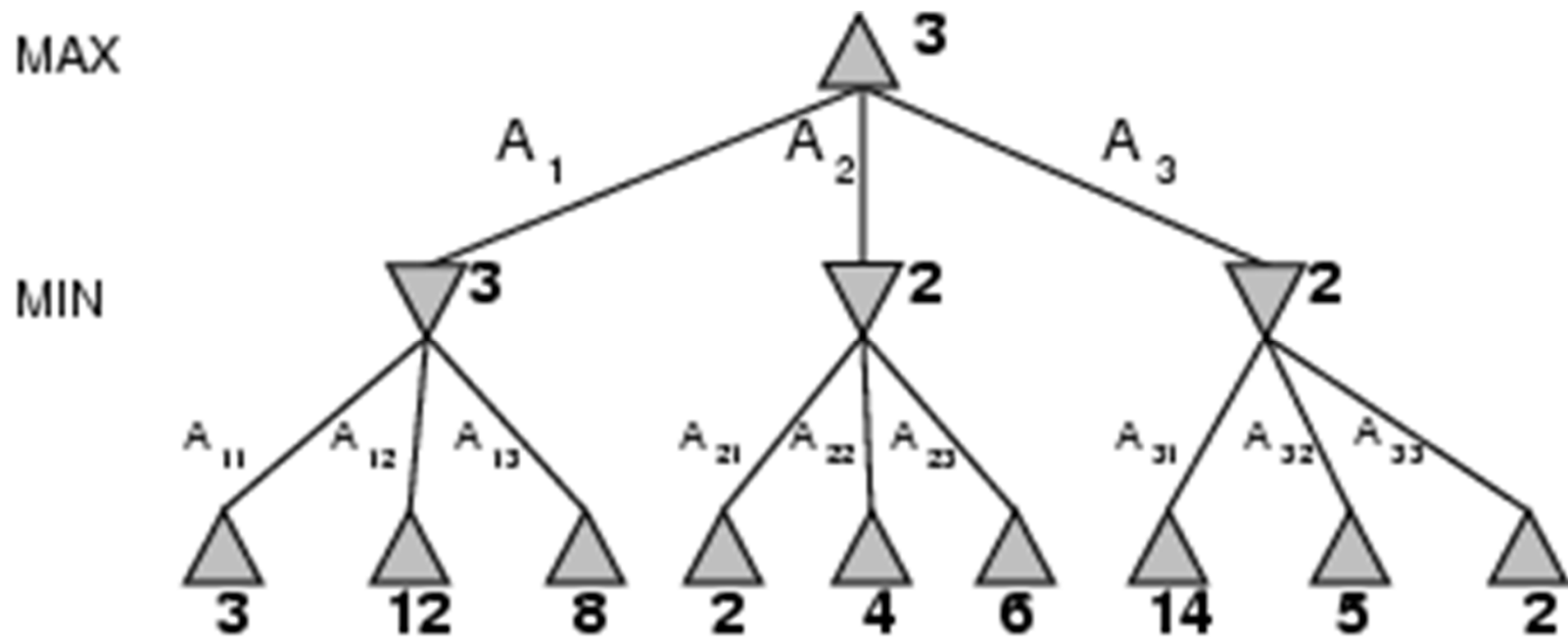


# Algoritmo Minimax

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



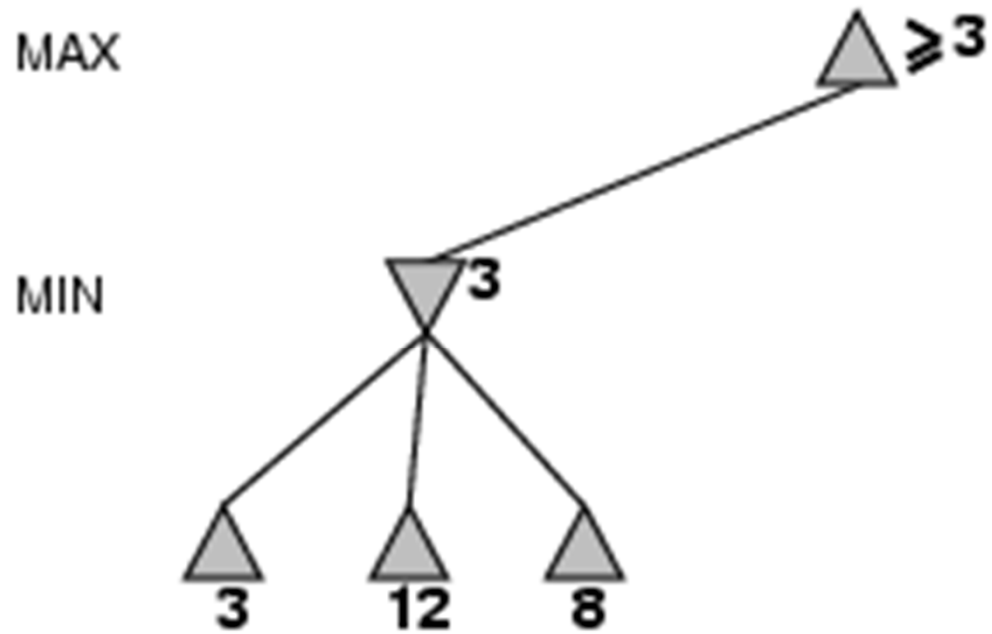
# Comentarios

- Sobre complejidad
  - Complejidad en tiempo
  - Complejidad en espacio.
- Variantes: NEGMAX.

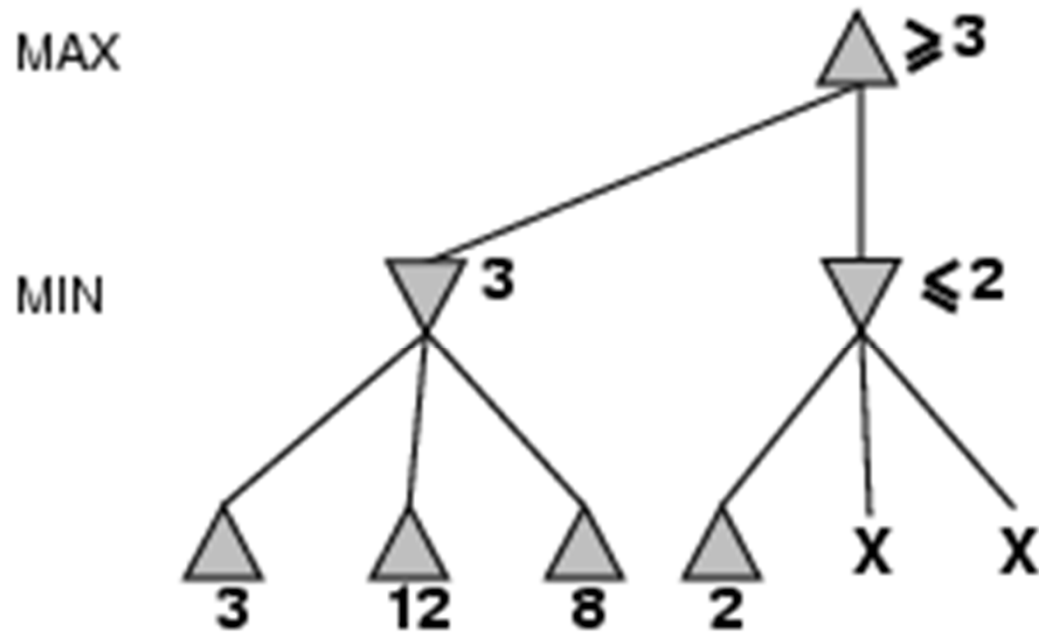
# Poda alfa-beta

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

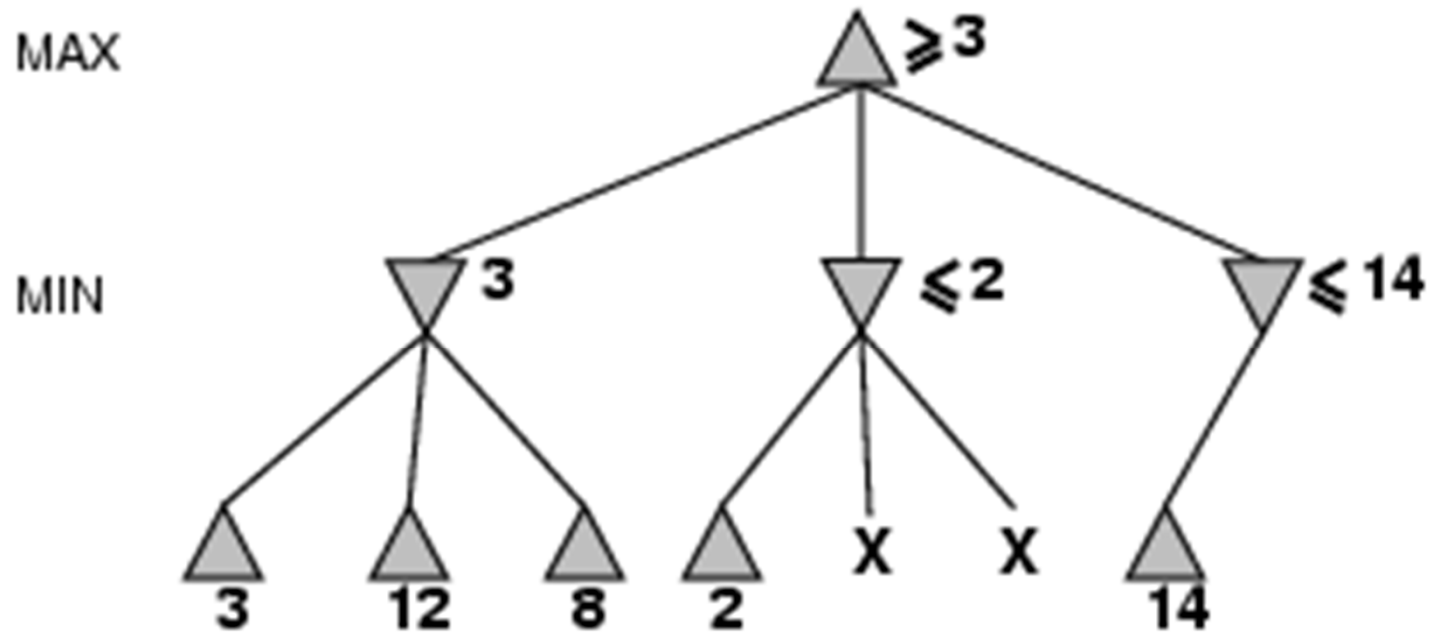
# Ejemplo poda



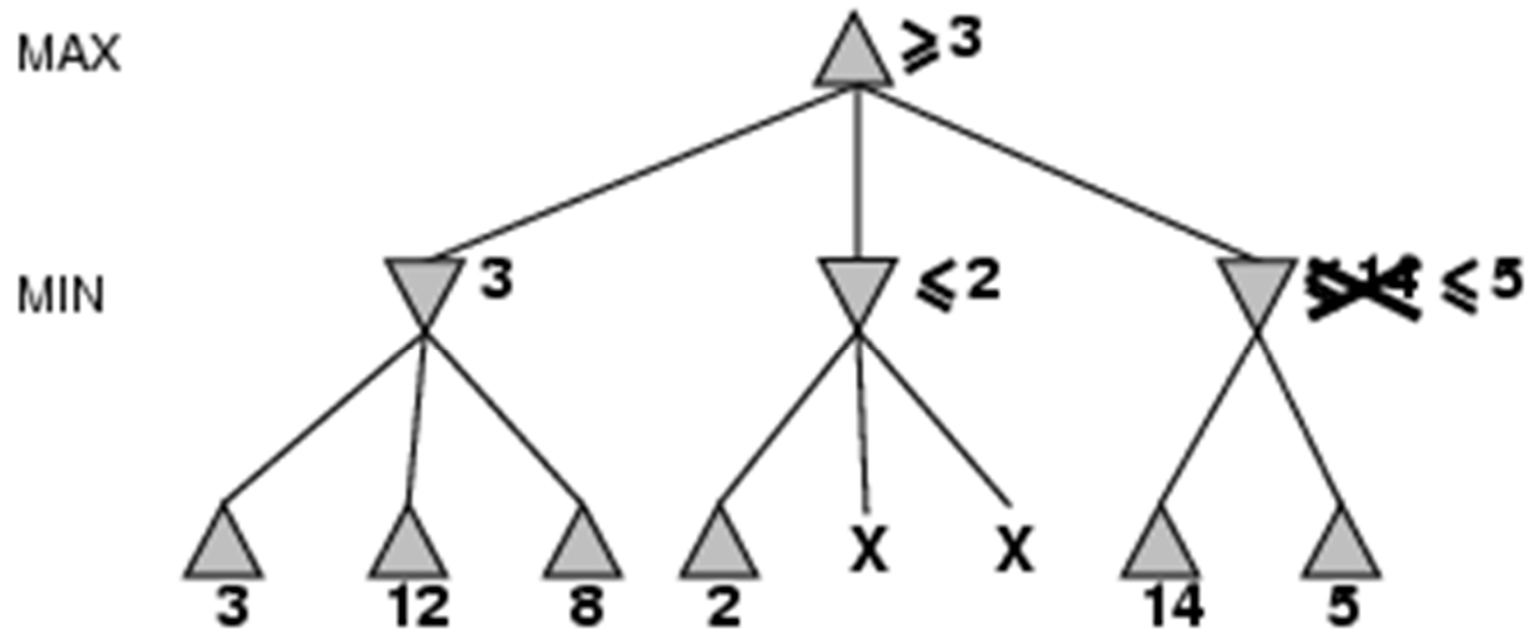
# Ejemplo poda



# Ejemplo poda

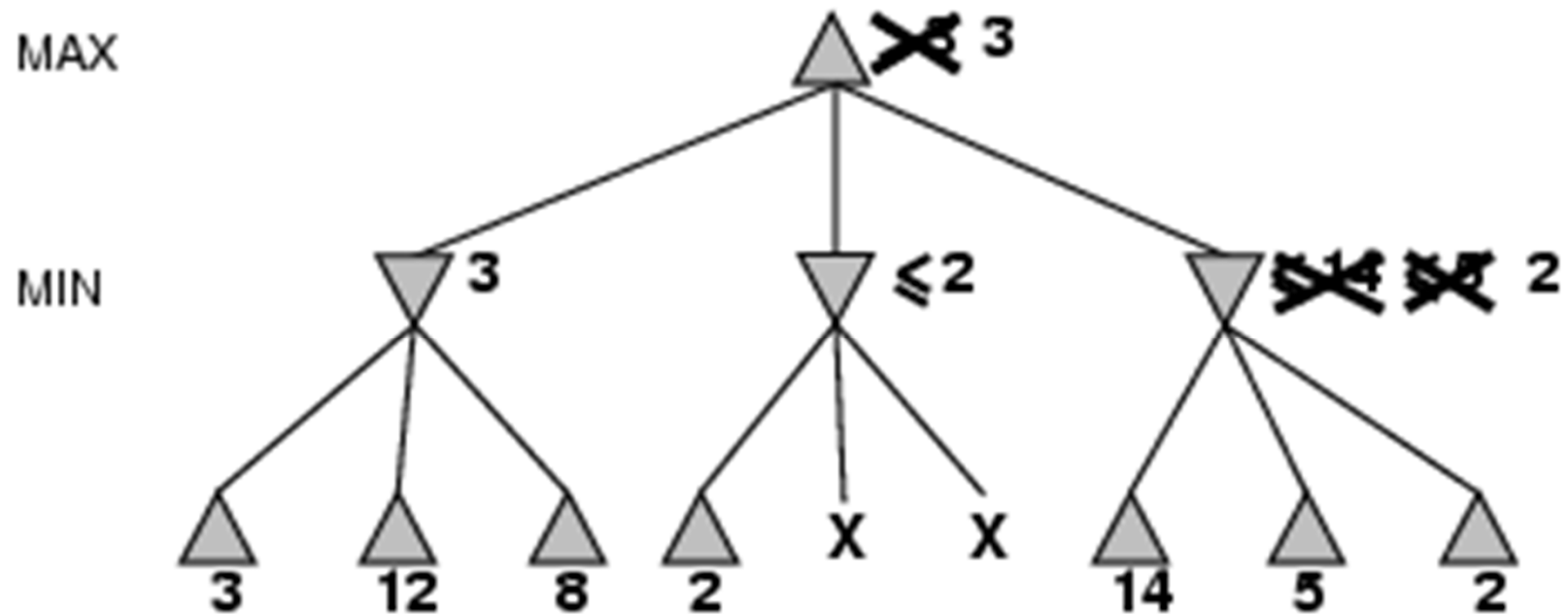


# Ejemplo poda





# Ejemplo poda



# Dos cotas

- Cada nodo va a tener dos variables asociadas  $\alpha$  y  $\beta$
- $\alpha$ : representa el **mejor valor** encontrado hasta el momento por los nodos MAX
  - Intuitivo: “variable auxiliar” usada por nodos MAX para calcular el máximo.
  - Inicialmente  $-\infty$
  - Es una cota inferior (**sólo puede crecer**)
  - Se actualiza en cada nodo MAX como resultado de evaluar sus hijos MIN.
- $\beta$ : representa el **mejor valor** encontrado hasta el momento por los nodos MIN
  - Intuitivo: “variable auxiliar” usada por nodos MIN para calcular el mínimo.
  - Inicialmente  $+\infty$
  - Es una cota superior (**solo puede decrecer**)
- Criterio de poda:
  - En cada nodo el intervalo  $[\alpha, \beta]$  se va estrechando conforme avanza la búsqueda.
  - Se poda cuando los valores se crucen.

# 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. NODO MAX
  1. Hacer  $\alpha \leftarrow \max(\alpha, V(J_k, \alpha, \beta))$ .
  2. Si  $\alpha \geq \beta$  devolver  $\beta$  (**¡criterio de poda!**); si no, continuar
  3. Si  $k=b$ , devolver  $\alpha$ ; si no, hacer  $k \leftarrow k+1$  y volver al paso 2.
3. NODO MIN
  1. Hacer  $\beta \leftarrow \min(\beta, V(J_k, \alpha, \beta))$ .
  2. Si  $\beta \leq \alpha$  devolver  $\alpha$  (**¡criterio de poda!**); si no, continuar
  3. Si  $k=b$ , devolver  $\beta$ ; si no, hacer  $k \leftarrow k+1$  y volver al paso 5.

# Comentarios

- Sobre complejidad.
- En resumen: es impracticable tratar de resolver un juego con un factor de ramificación y una profundidad “decentes”.
- Aun así la poda  $\alpha - \beta$  es la técnica de básica para la mayoría de las aplicaciones de juegos.

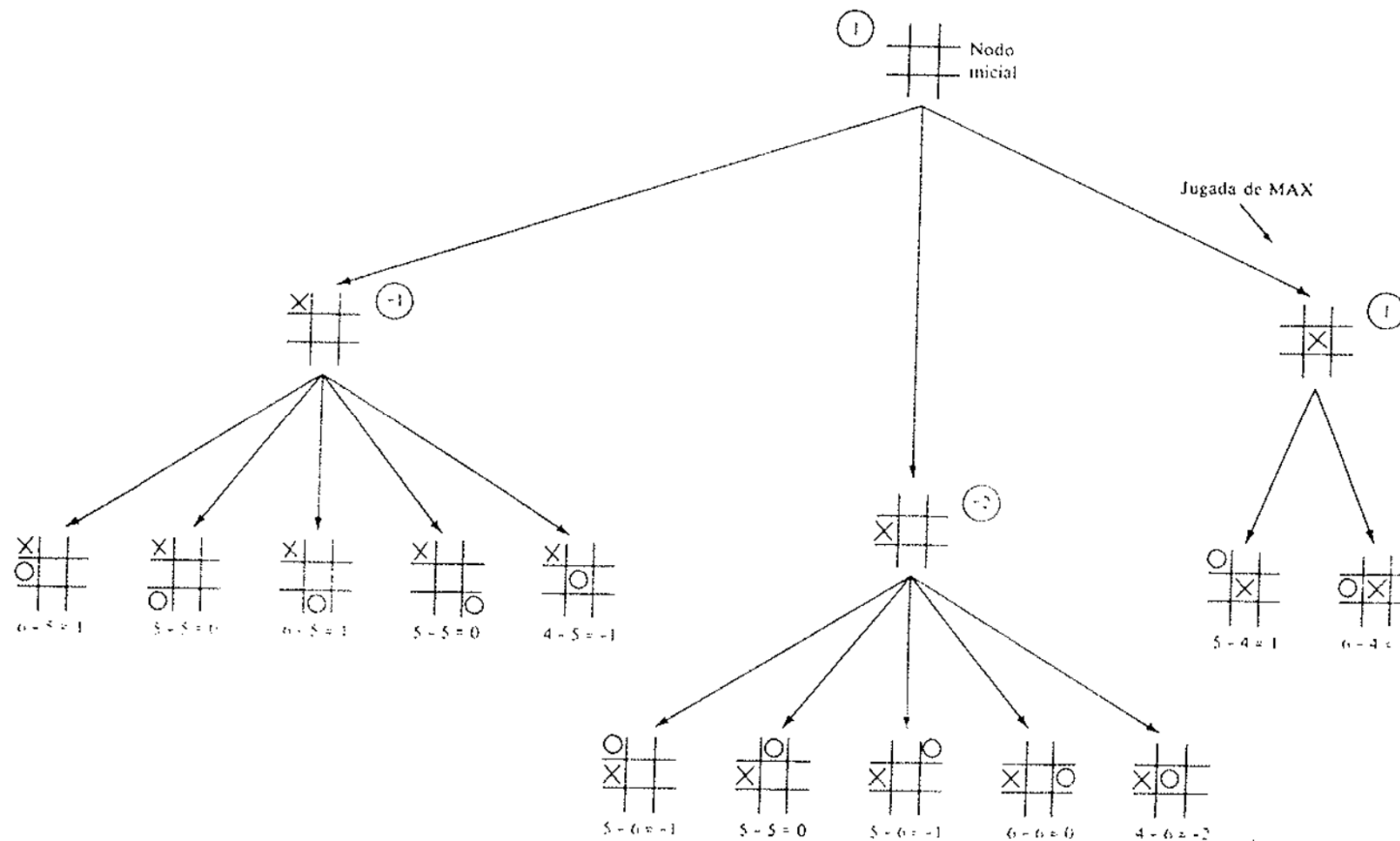
# Nuevo modelo de solución

- 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

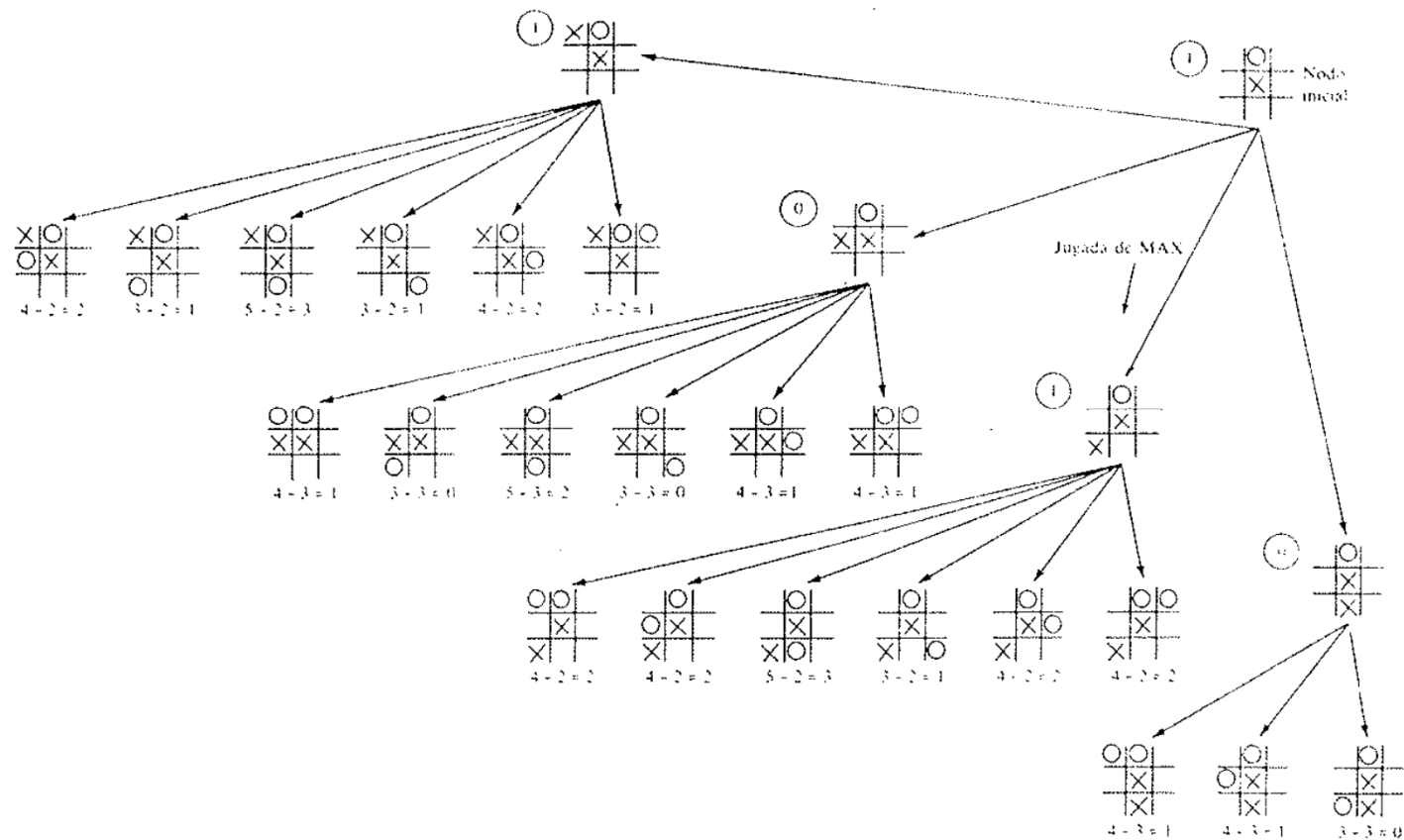
# El modelo básico para decisiones en tiempo real

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

# Ejemplo

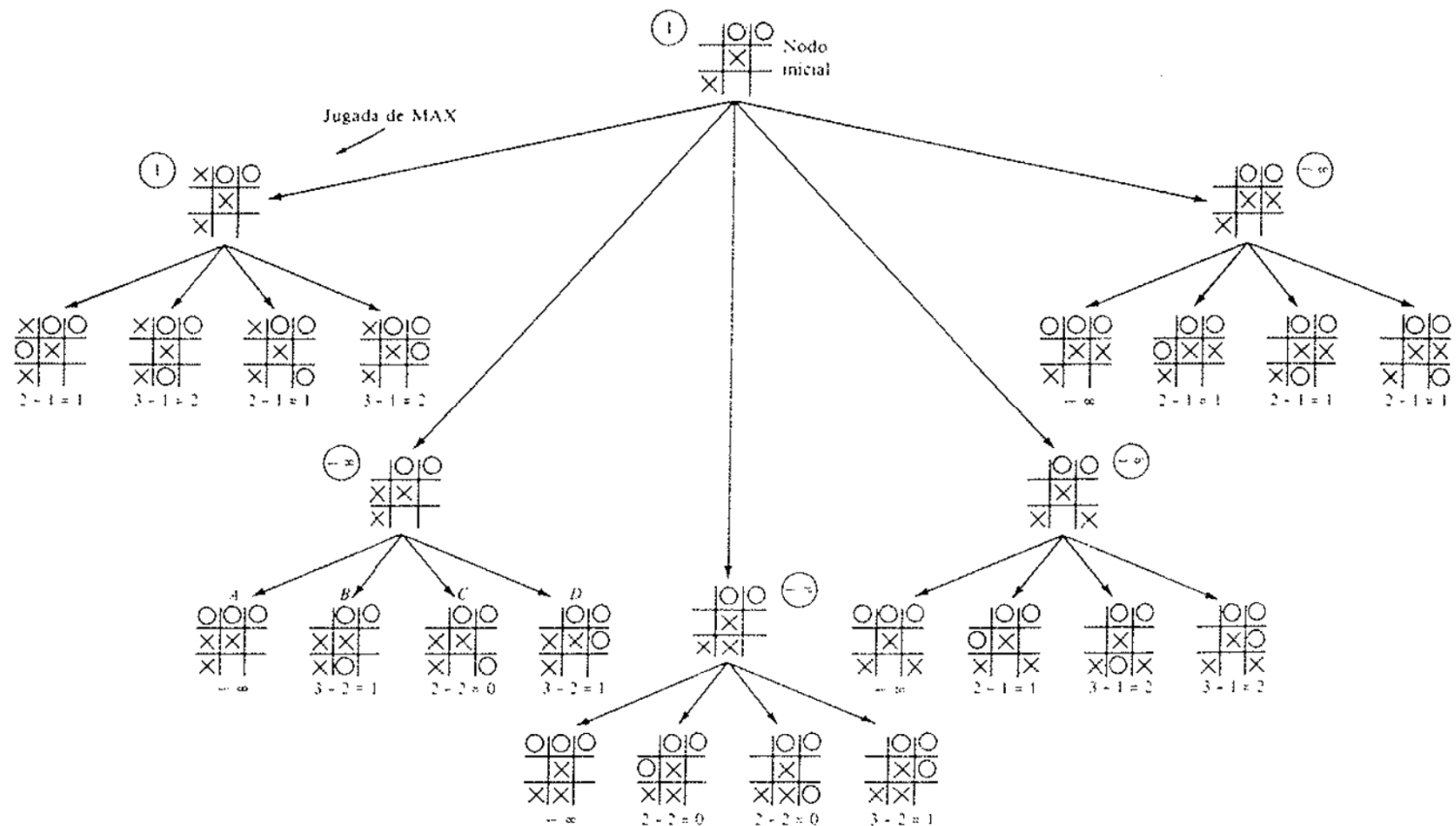


# Ejemplo

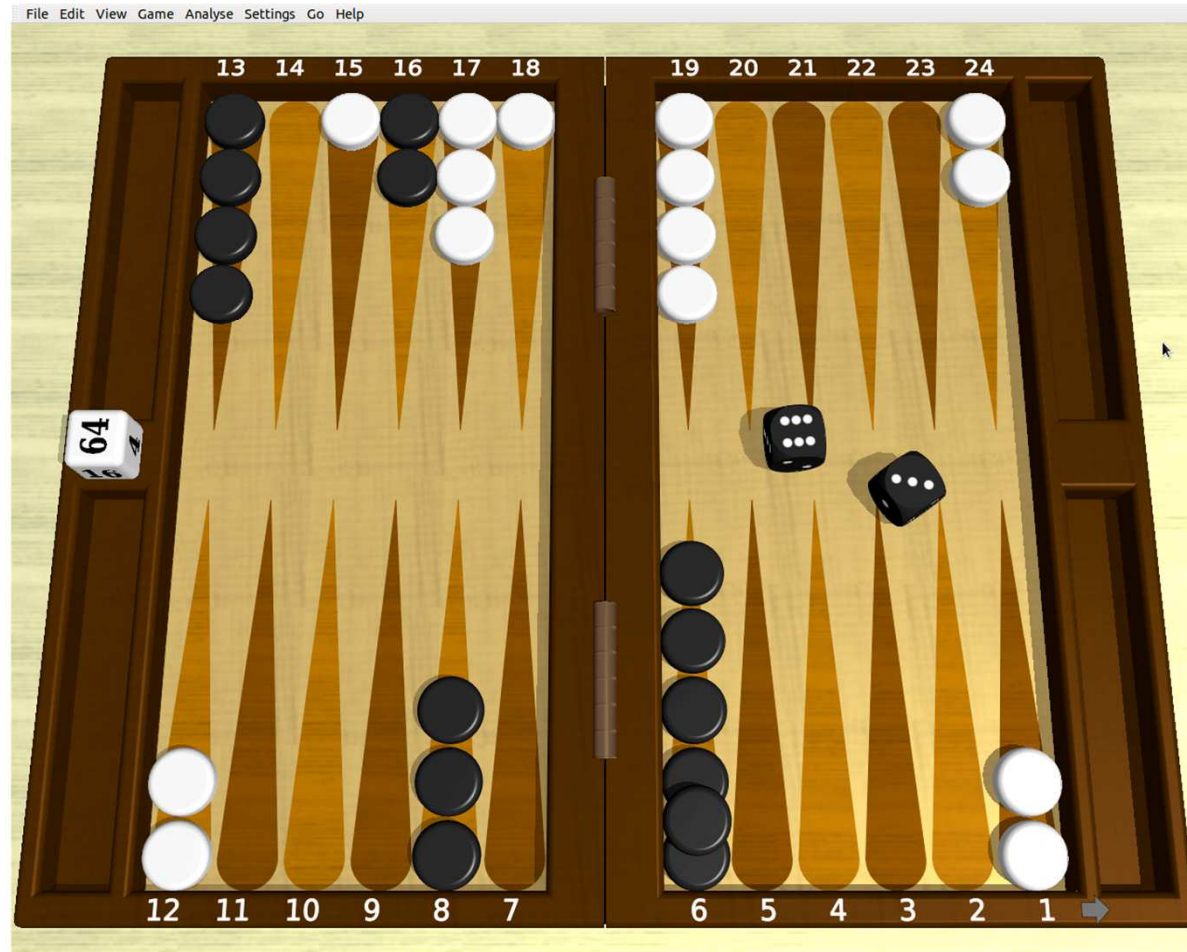




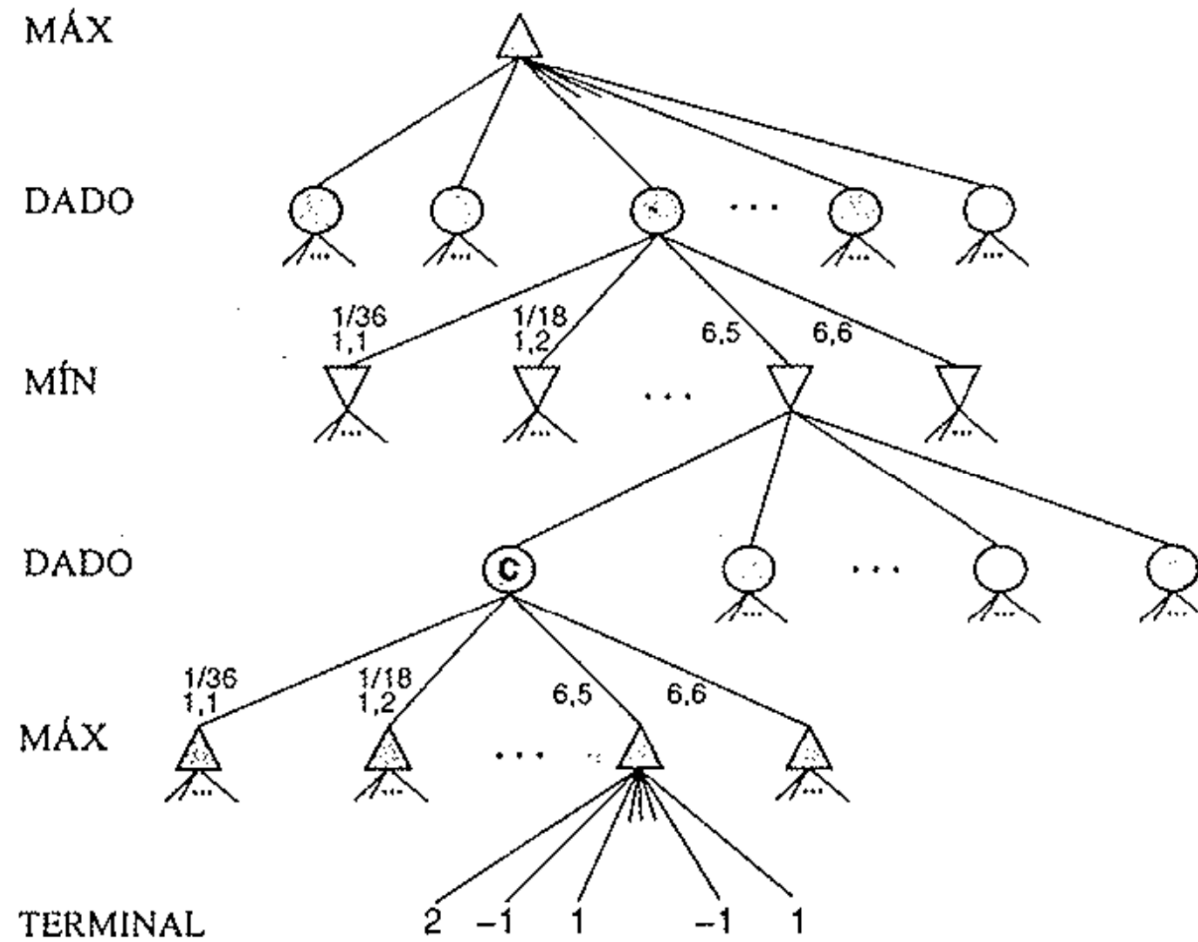
# Ejemplo



# Juegos en los que interviene un elemento aleatorio



# Modelo



# Algunos problemas

