

Puedo eliminar la publi de este documento con 1 coin

¿Cómo consigo coins? -



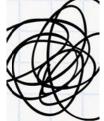
→ Plan Turbo: barato

Planes pro: más coins

pierdo espacio







Resumen Tema 4 – Búsqueda con adversario y juegos (IA)

Grado en Ingeniería Informática - UGR



- Comprender el funcionamiento de los algoritmos Minimax y Poda Alfa-Beta.
- Aplicarlos a juegos de adversario, especialmente los de dos jugadores con información perfecta.
- Introducir mejoras mediante heurísticas y nuevas estrategias como MCTS (Monte Carlo Tree Search).



🧩 1. Juegos bipersonales con información perfecta

Definición

- Juegos donde ambos jugadores conocen el estado completo del juego en todo momento.
- No hay azar ni información oculta.
- Se resuelven usando árboles de exploración y teoría de juegos.

Ejemplos

- Tres en raya
- **Damas**
- **Ajedrez**
- Juego de los palillos (quien retira el último pierde)

🎲 2. Teoría de Juegos

- Formaliza decisiones estratégicas entre varios agentes racionales.
- Fundada por Von Neumann y Morgenstern (1944).
- Incluye juegos **cooperativos** y **no cooperativos** (ej. dilema del prisionero).

Dilema del prisionero:

	No delatar	Delatar
No delatar	-2, -2	0, -10
Delatar	-10, 0	-5, -5

🕨 3. Árboles de juego y notación Minimax

- Representan todas las secuencias posibles de jugadas.
- Nodos MAX (jugador 1): intentan maximizar su recompensa.
- Nodos **MIN** (jugador 2): intentan minimizar la recompensa del contrario.
- Nodos terminales etiquetados según ganancia, derrota o empate para MAX.



4. Algoritmo MINIMAX

Objetivo:

Determinar el valor de un estado suponiendo que ambos jugadores juegan de forma óptima.

Funcionamiento:

- Si el nodo es **terminal**, se aplica la función de evaluación: V(J) = f(J)
- Si es MAX, se toma el máximo de sus hijos.
- Si es MIN, se toma el mínimo de sus hijos.



Fórmula:

MAX: $V(J) = max\{V(J1), V(J2), ..., V(Jb)\}$ MIN: $V(J) = min\{V(J1), V(J2), ..., V(Jb)\}$

۲ 5. Poda Alfa-Beta

Motivación:

Evitar la evaluación de ramas que no afectan a la decisión final ⇒ reduce complejidad sin perder precisión.

Cotas:

- α (alfa): mejor valor que puede garantizar MAX hasta el momento.
- **β (beta):** mejor valor que puede garantizar MIN hasta el momento.

Idea básica:

- Si un nodo MIN encuentra un valor ≤ α ⇒ se **poda** (MAX nunca permitirá llegar ahí).
- Si un nodo MAX encuentra un valor $\geq \beta \Rightarrow$ se **poda** (MIN nunca permitirá llegar ahí).

Complejidad:

- Sin poda: O(bp)O(b^p)O(bp)
- Con poda óptima: O(bp/2)O(b^{p/2})O(bp/2)

📐 6. Heurísticas

Se usan cuando no se puede explorar hasta los nodos terminales (por tiempo o complejidad).

Ejemplos:

- Ajedrez (Turing): B/N (balance de fichas)
- Damas (Samuel): función lineal ponderada $f(x) = \alpha 1 \cdot x 1 + \alpha 2 \cdot x 2 + \dots + \alpha n \cdot x n$





Puedo eliminar la publi de este documento con 1 coin

¿Cómo consigo coins? -



Plan Turbo: barato

Planes pro: más coins

pierdo espacio









Propagación:

- Se realiza hacia arriba desde las hojas con valores heurísticos.
- Se decide en base al valor estimado más favorable según sea MAX o MIN.

🧩 7. Mejoras en la búsqueda

Parámetros relevantes:

- Horizonte (profundidad)
- Evaluación hacia atrás
- Anchura y orden de la búsqueda

Tipos de programas:

- **Tipo A**: Generales, heurísticas globales.
- Tipo B: Juegan líneas específicas con alta profundidad.

🧘 8. Motores de juego avanzados

- **Stockfish**: motor determinista, usa alfa-beta y tablas de evaluación.
- Leela Chess Zero: aprendizaje por refuerzo (red neuronal).
- AlphaZero: aprendizaje completamente autónomo usando MCTS y redes neuronales profundas.



9. Monte Carlo Tree Search (MCTS)

Idea:

- Realiza simulaciones aleatorias completas desde el estado actual.
- Evalúa los nodos en base al **promedio de resultados**.



• No requiere heurísticas explícitas.

Fases:

- 1. Selección
- 2. Expansión
- 3. Simulación
- 4. Retropropagación

Ventajas:

- Funciona bien cuando el espacio de estados es muy grande.
- Utilizado en juegos como Go, donde alfa-beta falla por complejidad.

🔄 10. Juegos con azar

Cuando el juego incluye tiradas de dados u otras fuentes aleatorias:

- Se introducen nodos de azar en el árbol de búsqueda.
- Se modelan como árboles de decisión con nodos de esperanza matemática (valores ponderados por probabilidad).

Casos modernos

AlphaZero (DeepMind):

- Aprende desde cero sin datos previos.
- Usa MCTS + redes neuronales para aprender movimientos óptimos en ajedrez, go y shogi.

ChessBench:



- Entrenado con millones de partidas.
- Evaluado con Stockfish 16.
- Usado para investigación en evaluación de movimientos.

