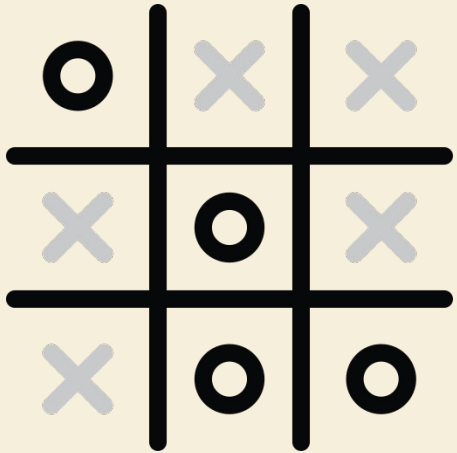
Tic-Tac-Toe TP1



Aliendo Marcos

JUEGOS DE SUMA CERO

- Entornos deterministas, totalmente observables en los cuales hay dos agentes cuyas acciones deben alternar y en los que los valores utilidad, al final de juego, son siempre iguales y opuestos.

DECISIONES ÓPTIMAS EN JUEGOS

- Jugadores MAX y MIN.
- Mueven por turno hasta que el juego termina.
- Al final del juego tenemos un ganador y un perdedor.

DEFINIENDO UN JUEGO

- El **estado inicial**, que incluye la posición del tablero e identifica al jugador que mueve.
- Una **función sucesor**, que devuelve una lista de pares (movimiento, estado), indicando un movimiento legal y el estado que resulta.
- Un **test terminal**, que determina cuándo se termina el juego. A los estados donde el juego se ha terminado se les llaman estados terminales.
- Una **función utilidad**, que da un valor numérico a los estados terminales. En este caso, el resultado es un triunfo, pérdida, o empate, con valores 1, -1, o.

Tic-Tac-Toe Funciones

Initial state

Devuelve el estado inicial del tablero

Player

Devuelve que jugador tiene el turno siguiente

Actions

Devuelve un conjunto de todas las acciones posibles en ese tablero

Result

Toma un tablero y una accion, y devuelve un nuevo estado sin modificar el tablero original.

Tic-Tac-Toe Funciones

Winner

Devuelve el ganador, si lo hay

Terminal

Indica si el juego ha terminado

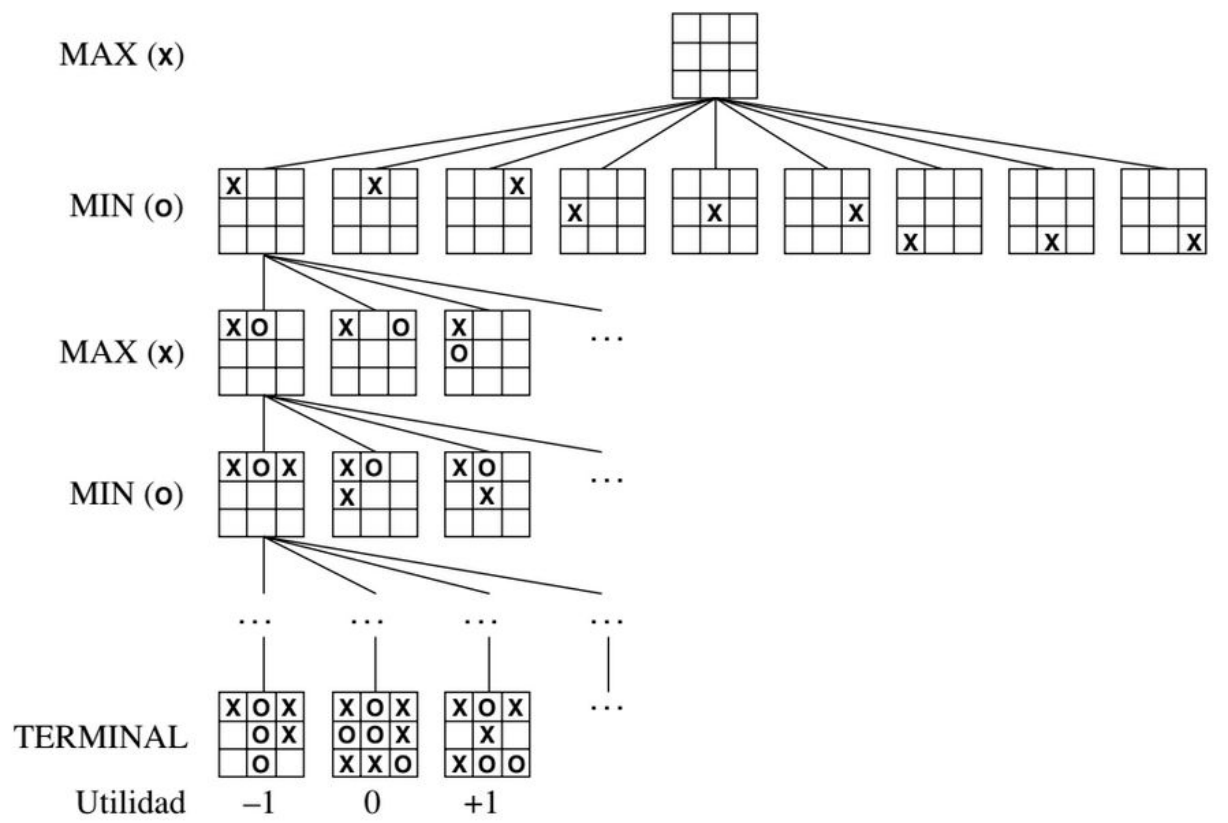
Utility

Calcula la utilidad del tablero

Minimax

Devuelve el movimiento óptimo

ARBOL DE JUEGOS



ESTRATEGIA OPTIMA

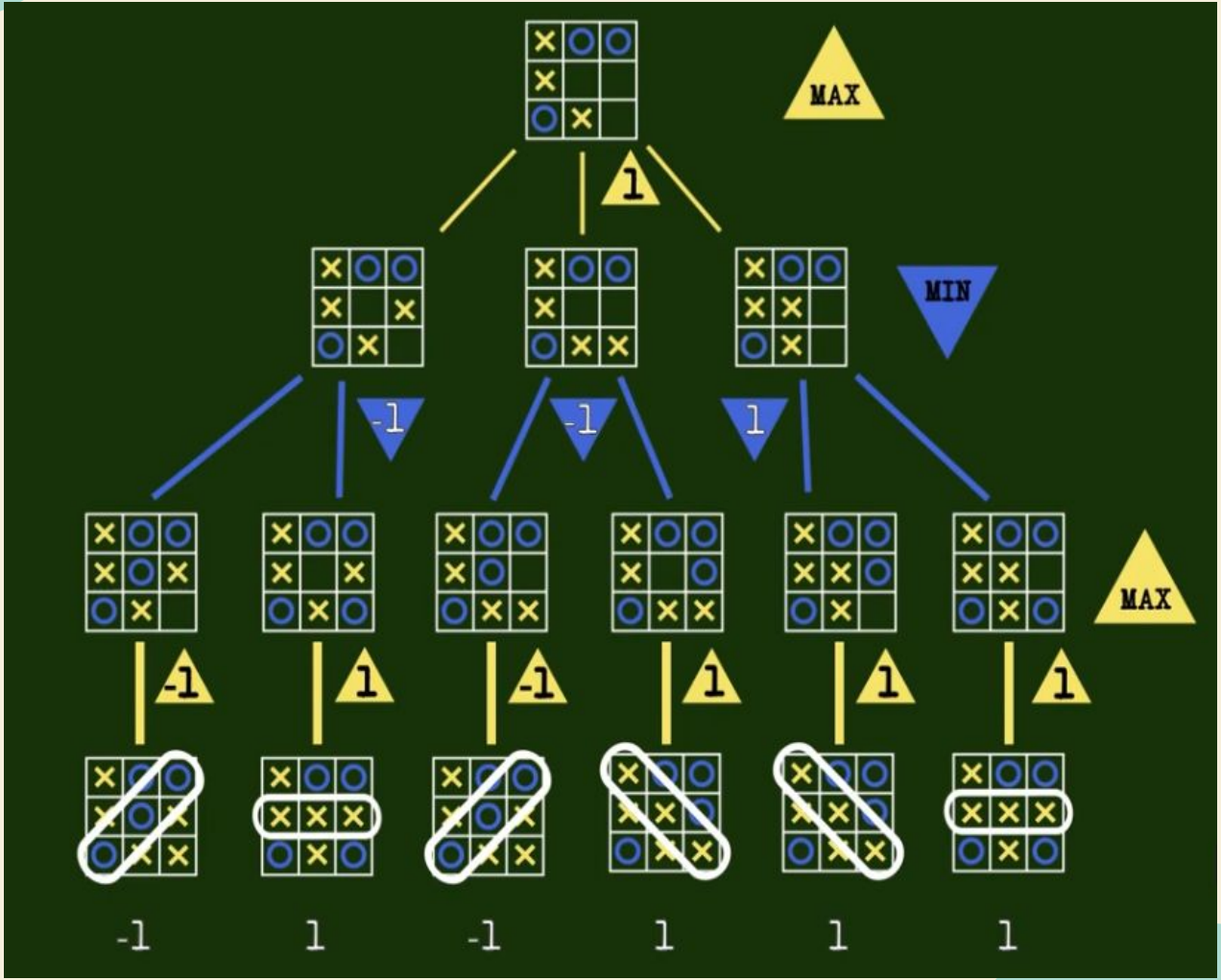
- El valor minimax de un nodo es la utilidad (para MAX) de estar en el estado correspondiente, asumiendo que ambos jugadores juegan óptimamente desde allí al final del juego. Obviamente, el valor minimax de un estado terminal es solamente su utilidad. Además, considerando una opción, MAX preferirá moverse a un estado de valor máximo, mientras que MIN prefiere un estado de valor mínimo.

ALGORITMO MINIMAX

```
funcion minimax(nodo, profundidad, esMaximizador):
    si profundidad = 0 o nodo es un nodo terminal entonces
        retornar el valor del nodo

    si esMaximizador entonces
        mejorValor = menos infinito
        para cada hijo de nodo hacer
            valor = minimax(hijo, profundidad - 1, Falso)
            mejorValor = max(mejorValor, valor)
        retornar mejorValor
    sino
        mejorValor = infinito
        para cada hijo de nodo hacer
            valor = minimax(hijo, profundidad - 1, Verdadero)
            mejorValor = min(mejorValor, valor)
        retornar mejorValor

// Uso del algoritmo
valor = minimax(raiz, profundidadInicial, Verdadero)
```

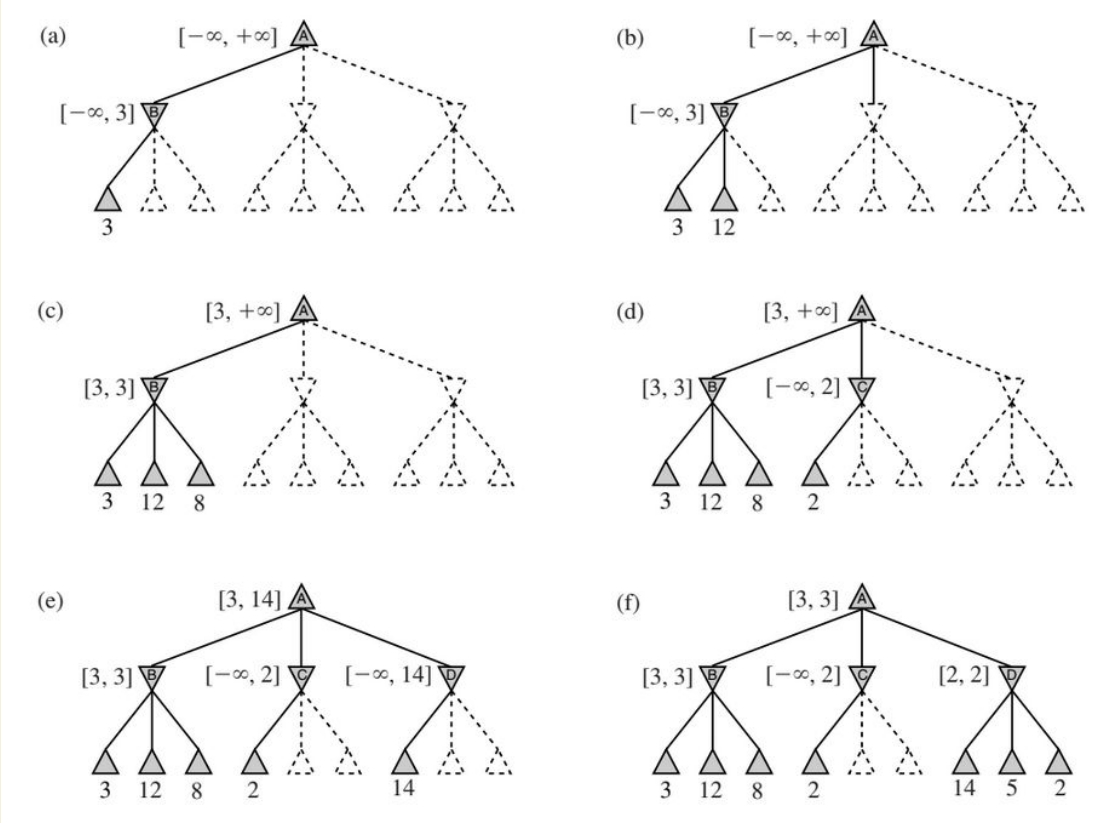

MAX
1
MIN
-1
-1
1
-1
1
-1
1
1
1
MAX
-1
1
-1
1
1
1


MARAVILLOSA JUGADA

PODA ALFA-BETA

- El problema de la búsqueda minimax es que el número de estados que tiene que examinar es exponencial en el número de movimientos.
- La jugada es que es posible calcular la decisión minimax correcta sin mirar todos los nodos en el árbol de juegos.

PODA ALFA-BETA



GRACIAS!