



UNIVERSIDAD NACIONAL MAYOR DE SAN MARCOS
FACULTAD DE INGENIERÍA DE SISTEMAS E INFORMÁTICA
ESCUELA ACADÉMICA PROFESIONAL DE INGENIERÍA DE SISTEMAS

CURSO: Inteligencia Artificial

TÍTULO: Métodos de búsqueda para juegos humano-máquina

INTEGRANTES:

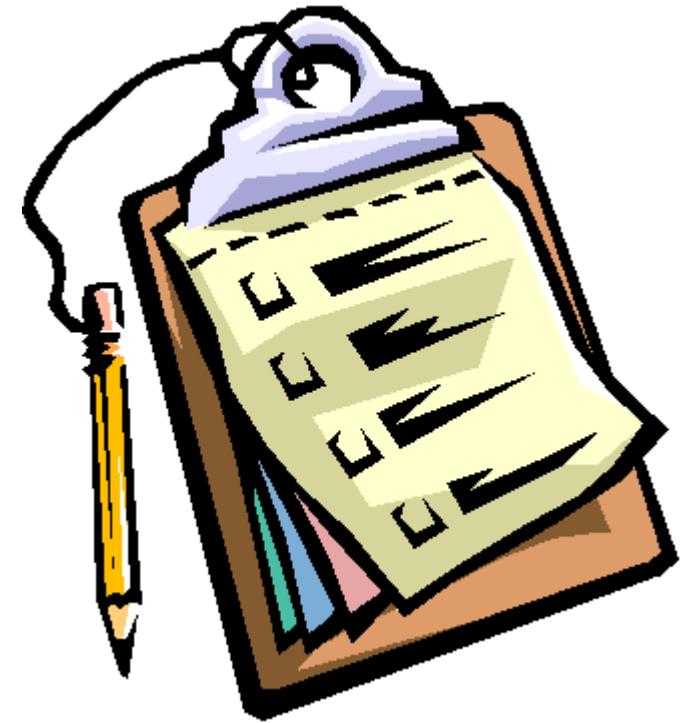
- Collao Aldave, Max Jairo
- Cuestas Gomez, Julyño Andres
- Rojas Orihuela, Kevin Gustavo
- Tapia Huaroto, John Eduardo
- Yupanqui Garcia, Andy Ander



UNIVERSIDAD NACIONAL MAYOR DE SAN MARCOS
FACULTAD DE INGENIERÍA DE SISTEMAS E INFORMÁTICA
ESCUELA ACADÉMICA PROFESIONAL DE INGENIERÍA DE SISTEMAS

AGENDA

- Estrategias de juego de máquina:
- Estrategia no determinístico
- Estrategia primero el mejor
- Estrategia min-max
- Estrategia mejor diferencia de utilidades
- Algoritmo min-max y alfa-beta





Estrategia No-Determinístico

Este tipo de estrategia se requiere en los juegos un nivel de dificultad: PRINCIPIANTE. Consiste en seleccionar en forma aleatoria una de las posibles jugadas.

Estas estrategias no deterministas pueden explorar y aprender por su propia cuenta, además promueven un comportamiento emergente o que no siga instrucciones explícitas.

Por ejemplo de un comportamiento no determinista es cuando un jugador-máquina que aprende a adaptarse a las tácticas bélicas de un jugador.



Estrategia Primero el Mejor

Consiste en un algoritmo que hace su proceso de búsqueda en un grafo de tipo O , ya que todos sus ramales representan una alternativa de solución. Para su operación, el algoritmo necesita dos listas de nodos y una función heurística que estime los méritos de cada nodo que se genere:

ABIERTOS - Variable que contiene los nodos que han sido generados. La función heurística ha sido aplicada a ellos pero no se han generado sus sucesores.

CERRADOS - Variable que contiene los nodos que han sido examinados. Es necesario tener esta información, para que la búsqueda sea en un grafo y no en un árbol.

FUNCIÓN HEURÍSTICA - Permite que el algoritmo busque primero por senderos que son o parecen más prometedores



Estrategia MIN-MAX

Es un método básico para recorrer un árbol y seleccionar movimiento más prometedor.

Es un método de decisión para *minimizar* la pérdida *máxima* esperada en juegos con adversario y con información perfecta. Minimax es un algoritmo recursivo.

El funcionamiento de Minimax puede resumirse como elegir el mejor movimiento para ti mismo suponiendo que tu contrincante escogerá el peor para ti.



Estrategia mejor diferencia de utilidades

Se selecciona la jugada que genera mayor diferencia de utilidades entre la jugada de la máquina y la correspondiente mejor jugada para el humano.



Algoritmo MIN-MAX

Pasos del algoritmo Minimax:

- Generación del árbol de juego. Se generarán todos los nodos hasta llegar a un estado terminal.
- Cálculo de los valores de la función de utilidad para cada nodo terminal.
- Calcular el valor de los nodos superiores a partir del valor de los inferiores. Alternativamente se elegirán los valores mínimos y máximos representando los movimientos del jugador y del oponente, de ahí el nombre de Minimax.
- Elegir la jugada valorando los valores que han llegado al nivel superior.



Algoritmo MIN-MAX

función Decisión-Minimax(*estado*) **devuelve** una acción

variables de entrada: *estado*, estado actual del juego

$v \leftarrow \text{Max-Valor}(\text{estado})$

devolver la acción de *Sucesores*(*estado*) con valor v

función Max-Valor(*estado*) **devuelve** un valor utilidad

si Test-Terminal(*estado*) **entonces devolver** Utilidad (*estado*)

$v \leftarrow -\infty$

para un s en *Sucesores*(*estado*) **hacer**

$v \leftarrow \text{Max}(v, \text{Min-Valor}(s))$

devolver v

función Min-Valor(*estado*) **devuelve** un valor utilidad

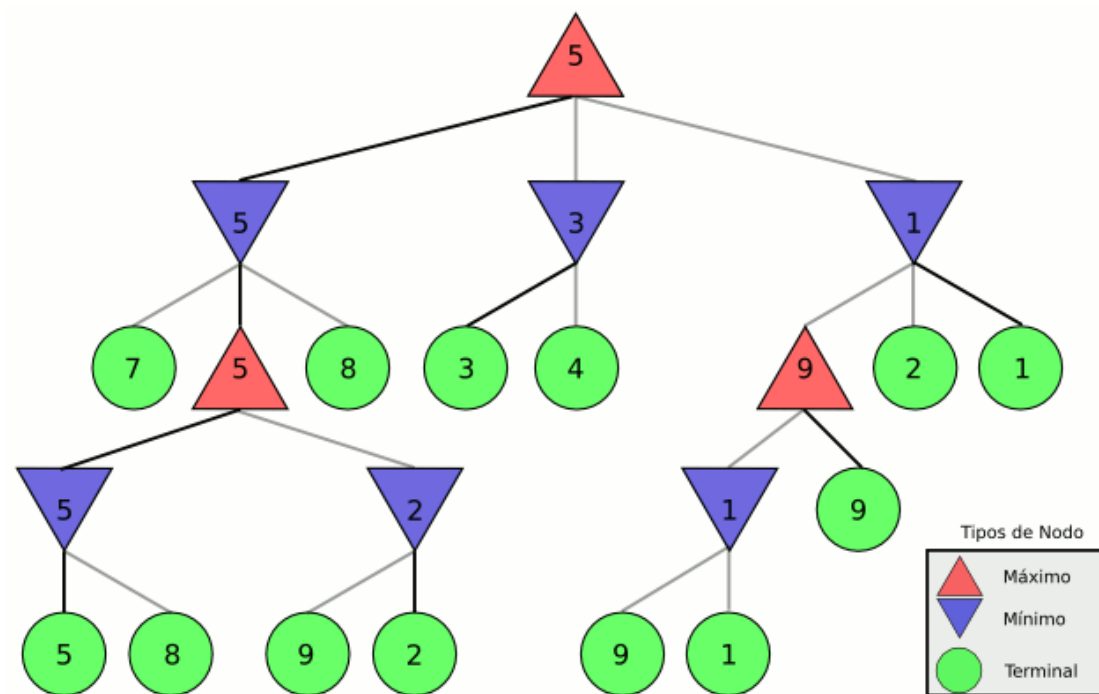
si Test-Terminal(*estado*) **entonces devolver** Utilidad (*estado*)

$v \leftarrow \infty$

para un s en *Sucesores*(*estado*) **hacer**

$v \leftarrow \text{Min}(v, \text{Max-Valor}(s))$

devolver v





Estrategia alfa-beta

La búsqueda minimax es primero en profundidad, por ello en cualquier momento sólo se deben considerar los nodos a lo largo de un camino en el árbol.

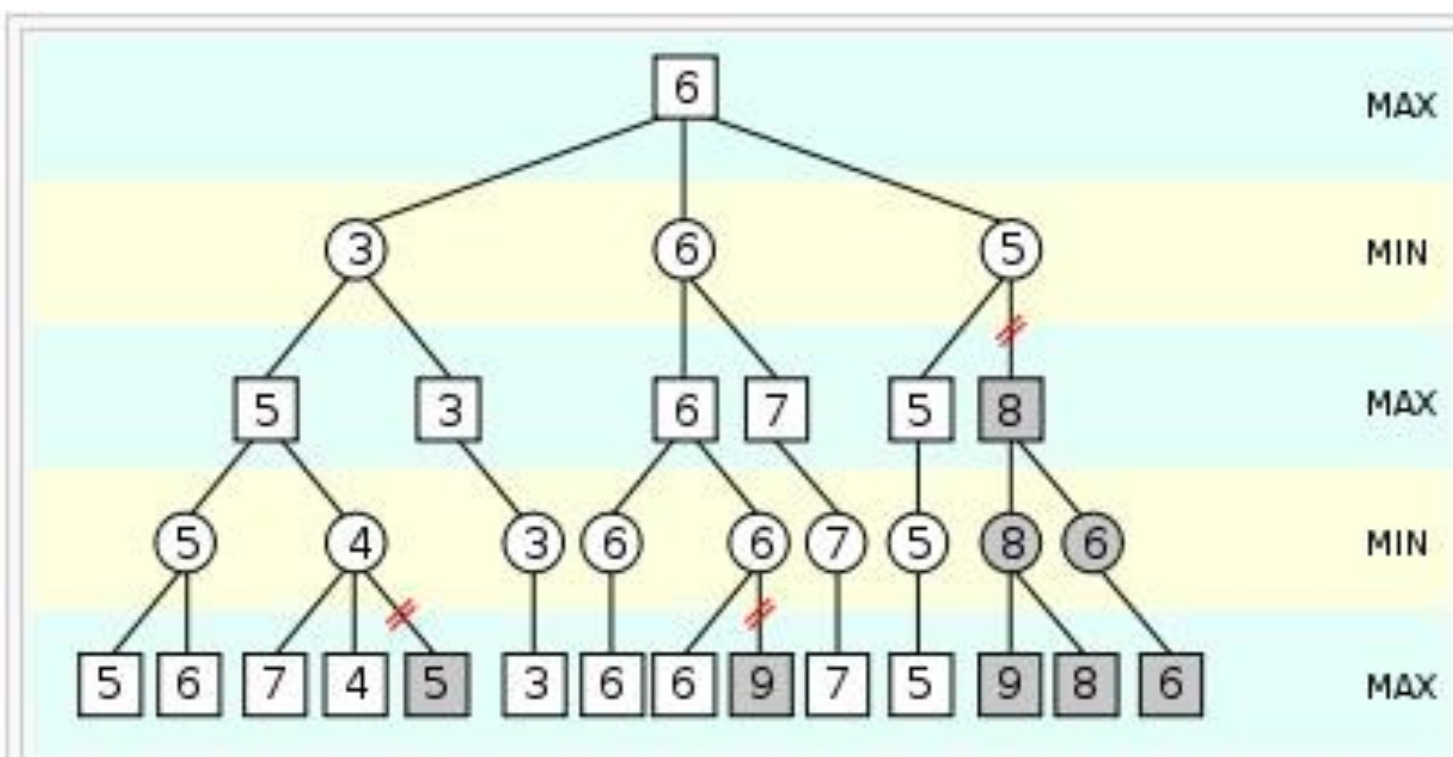
La poda alfa-beta toma dicho nombre de la utilización de dos parámetros que describen los límites sobre los valores hacia atrás que aparecen a lo largo de cada camino.

- α es el valor de la mejor opción hasta el momento a lo largo del camino para MAX, esto implicará por lo tanto la elección del valor más alto
- β es el valor de la mejor opción hasta el momento a lo largo del camino para MIN, esto implicará por lo tanto la elección del valor más bajo.

Esta búsqueda alfa-beta va actualizando el valor de los parámetros según se recorre el árbol. El método realizará la poda de las ramas restantes cuando el valor actual que se está examinando sea peor que el valor actual de α o β para MAX o MIN, respectivamente.



Poda alfa-beta



La figura muestra un ejemplo de la poda alfa-beta. Cada nivel representa la jugada de los jugadores MAX y MIN, que tendrán que definir un valor α o β respectivamente.





Algoritmo alfa-beta

```
función alfa-beta(nodo //en nuestro caso el tablero, profundidad,  $\alpha$ ,  $\beta$ , jugador)
  si nodo es un nodo terminal o profundidad = 0
    devolver el valor heurístico del nodo
  si jugador1
    para cada hijo de nodo
       $\alpha := \max(\alpha, \text{alfa-beta}(\text{hijo}, \text{profundidad}-1, \alpha, \beta, \text{jugador2}))$ 
      si  $\beta \leq \alpha$ 
        romper (* poda  $\beta$  *)
    devolver  $\alpha$ 
  si no
    para cada hijo de nodo
       $\beta := \min(\beta, \text{alfa-beta}(\text{hijo}, \text{profundidad}-1, \alpha, \beta, \text{jugador1}))$ 
      si  $\beta \leq \alpha$ 
        romper (* poda  $\alpha$  *)
    devolver  $\beta$ 
```

```
(* Llamada inicial *)
alfa-beta(origen, profundidad, -infinito, +infinito, jugador_deseado)
```