

# *Makarena*: Deducción evolutiva de funciones de evaluación aplicadas a *Minimax*

Ignacio Slater Muñoz

Departamento de Ciencias de la Computación

Universidad de Chile

Santiago, Chile

3 de abril de 2022

`ignacio.slater@ug.uchile.cl`

## 1. Estado del arte

### 1.1. Función de evaluación

**Definición 1.1 (Recurso).** Se le dirá recurso a cada elemento en el juego que acerque a un jugador a una jugada óptima.

Se le dirá **función de evaluación**[5]  $v_i(s)$  a la función que determina la cantidad de recursos que tiene el jugador  $i$  en el estado  $s$ . A su vez, el **estado del juego** se define como el par  $a_i, a_{-i}$  que representa las acciones de los jugadores que llevan a una configuración específica del juego (un ejemplo de configuración puede ser la posición y cantidad de piezas en el tablero).

### 1.2. Juegos de suma cero

Los **juegos de suma cero**[7] (*zero sum games*) son una representación matemática usada en *teoría de juegos* y *teoría económica* para describir una situación donde existen dos «jugadores» que se enfrentan entre sí. Un *juego es de suma cero* si la ventaja que lleva un jugador es igual a la pérdida del otro.

Los *juegos de suma cero* son un tipo de juegos de suma constante donde la suma de ganancia y pérdida es igual a cero.

En un juego de suma cero, todos los jugadores perciben la misma ganancia de cada recurso. Para entender esto podemos considerar un juego con tres estados: empate, ganador y perdedor, con puntajes 0, 1 y -1 respectivamente.

- **Empate:** Ningún jugador tiene ventaja; ambos jugadores tienen 0 puntos.
- **Ganador:** El jugador ganó, por lo que tiene 1 punto; el otro jugador pierde así que tiene -1 puntos.
- **Perdedor:** El jugador perdió, por lo que tiene -1 puntos; el otro jugador ganó así que tiene 1 punto.

De esta forma, para todo estado del juego, la suma de los puntajes es 0.

### 1.3. *Minimax*

*Minimax*[6, 1, 2] (*MM*) es un algoritmo de decisión utilizado en múltiples ámbitos para minimizar la pérdida posible para el «peor caso» (*maximum loss*). Se formuló originalmente para juegos de suma cero de  $n$  jugadores, pero se ha extendido a juegos más complejos y a problemas de decisión con incertidumbre.

**Definición 1.2 (Valor *Maximin*).** El valor *maximin* de un estado del juego es la máxima cantidad de recursos que puede obtener un jugador, sin conocer las jugadas de su oponente. Formalmente:

$$\underline{v}_i = \max_{a_i} \min_{a_{-i}} v_i(a_i, a_{-i})$$

Donde:

- $i$  es el índice del jugador actual.
- $-i$  representa a todos los jugadores oponentes.
- $a_i$  es la jugada del jugador  $i$ .
- $a_{-i}$  son las jugadas de todos los oponentes.
- $v_i$  es la función de evaluación del estado del juego en el turno del jugador  $i$ .

**Definición 1.3 (Valor *Minimax*).** El valor *minimax* de un estado del juego es la mínima cantidad de recursos que un oponente puede forzar al jugador a perder, sin saber las jugadas del jugador.

Formalmente:

$$\overline{v}_i = \min_{a_{-i}} \max_{a_i} v_i(a_i, a_{-i})$$

**Definición 1.4.** Se define la función *MM* para un estado del juego  $s$  como:

$$MM_i(s) = \begin{cases} v_i(s) & \text{si } s = F \\ \underline{v}_i(s) & \text{si es el turno del jugador } i \\ \overline{v}_i(s) & \text{si es el turno del oponente } -i \end{cases}$$

Con  $F$  representando el estado final del juego.

**Definición 1.5 (Algoritmo *Minimax*).** Sea  $t$  un árbol de estados del juego, donde cada nodo del árbol es un estado del juego evaluado con la función de evaluación  $v_i$ , y donde cada arco del árbol es una jugada de un jugador (para simplificar diremos que cada jugada representa un turno y que los jugadores toman turnos alternados, primero  $i$  y luego  $-i$ ). Luego, cada nodo tendrá tantos hijos como jugadas pueda realizar cada jugador en su turno.

Se define el algoritmo *minimax* para un árbol de estados  $t$  como:

```
fun minimax(t: StateTree, player: Player in [i, -i]): Double =  
  if (t.isTerminal()) {  
    t.value  
  } else if (player == -i) {  
    t.children.maxOf { child -> minimax(child, i) }  
  } else {  
    t.children.minOf { child -> minimax(child, -i) }  
  }
```

## 2. Pregunta de investigación

Dada una representación del estado de un juego. ¿Es posible definir un algoritmo que encuentre una función de evaluación del estado del juego para la toma de decisiones del algoritmo *Minimax*?

## 3. Hipótesis

Es posible utilizar *programación genética* para derivar la función de evaluación como un árbol de sintaxis abstracta.

## Referencias

- [1] J v Neumann. «Zur theorie der gesellschaftsspiele». En: *Mathematische annalen* 100.1 (1928), págs. 295-320.
- [2] Ky Fan. «Minimax theorems». En: *Proceedings of the National Academy of Sciences of the United States of America* 39.1 (1953), pág. 42.
- [3] Michael Maschler, Eilon Solan y Shmuel Zamir. *Game Theory*. Cambridge University Press, 2013. DOI: 10.1017/CB09780511794216.
- [4] Magarena Community. *magarena/magarena: Magarena is a single-player fantasy card game played against a computer opponent*. URL: <https://github.com/magarena/magarena> (visitado 24-03-2022).
- [5] Wikipedia, the free encyclopedia. *Evaluation function*. URL: [https://en.wikipedia.org/wiki/Evaluation\\_function](https://en.wikipedia.org/wiki/Evaluation_function) (visitado 15-03-2022).
- [6] Wikipedia, the free encyclopedia. *Minimax*. URL: <https://en.wikipedia.org/wiki/Minimax> (visitado 30-03-2022).
- [7] Wikipedia, the free encyclopedia. *Zero-sum game*. URL: [https://en.wikipedia.org/wiki/Zero-sum\\_game](https://en.wikipedia.org/wiki/Zero-sum_game) (visitado 30-03-2022).