# Implementación de Tic Tac Toe mediante árboles de decisiones

## Tic Tac Toe implementation through decision trees.

Autor 1: Moisés Alfonso Ducuara Ospina
Autor 2: Javier Mauricio Gil Cardona
*Universidad Tecnológica de Pereira, Risaralda, Colombia*
Correo-e: moises.ducuara@utp.edu.co
Correo-e: mauriciogi@utp.edu.co

***Resumen*— En Tic Tac Toe se puede implementar los dos o un solo jugador automatizado usando inteligencia artificial, mediante árboles de decisiones, este método es capaz de tomar tomar decisiones con gran precisión. Este sistema de aprendizaje supervisado aplica la estrategia de "divide y vencerás# para hacer la clasificación, implementando métodos y técnicas para la realización de procesos inteligentes, representando así el conocimiento y el aprendizaje, con el propósito en este caso de automatizar el juego de Tic Tac Toe.**

***Palabras clave*— Tic Tac Toe, árboles de decisiones, aprendizaje supervisado, procesos inteligentes, inteligencia artificial.**

***Abstract*— In Tic Tac Toe you can implement the two or a single player automated using artificial intelligence, through decision trees, this method is capable of making decisions with great precision. This supervised learning system applies the strategy of "divide and conquer # to make the classification, implementing methods and techniques to carry out intelligent processes, thus representing knowledge and learning, with the purpose in this case of automating the game of Tic Tac Toe.**

***Key Word* — Tic Tac Toe, decision trees, supervised learning, intelligent processes, artificial intelligence.**

## I. INTRODUCCIÓN

El juego Tic Tac Toe o también conocido como triqui o tres en línea es un juego tan conocido como lo puede ser el ajedrez, consta de 2 jugadores y un tablero con 9 cuadrículas las cuales cada jugador por turno va a ir llenando con su respectiva figura las cuales pueden ser "X" que es el jugador que inicia siempre la partida o "O". El ganador se define cuando alguno de los dos jugadores logre poner 3 de sus figuras alineadas en cualquier sentido, ya sea horizontal, vertical e incluso diagonal, si ninguno de los dos jugadores logra formar la secuencia de 3 figuras al llenarse todo el tablero, se decreta que es un empate en la partida esto se ve más claro en la **Figura 1**.

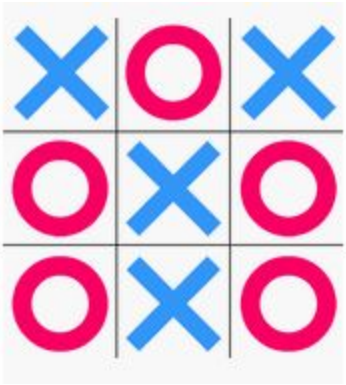

**FIgura 1.**

El juego tiene una lógica básica la cual es que el primero que empieza, ósea el jugador que le corresponde la "X" es el que ataca y el jugador al que le corresponde el "O" es aquél que se defiende. Lo que se tiene que tener presente es que se va a realizar un juego en el cual una persona va a poder jugar contra una máquina y la idea es que esta máquina haga las mejores jugadas posibles tanto cuando sea "X" o cuando sea "O", para esto hay que buscar métodos en los cuales el sistema tenga almacenadas todas las jugadas posibles de una manera eficiente.

Si se quisiera resolver esto de la manera más fácil que es a pedal; es decir, se haría todo el árbol para una búsqueda exhaustiva lo cual tomaría demasiado tiempo en realizar y no sería muy eficiente y óptimo a la hora de usar recursos en un dispositivo ya que las búsquedas se tornarán largas y abrumadoras para el sistema y la idea no es esa, la idea es

Fecha de Recepción: ""
Fecha de Aceptación:

construirlo de una manera que funcione igualmente bien, pero manteniendo un estándar óptimo de rendimiento en un equipo.

Y esto se hace también con búsquedas exhaustivas, pero con funciones limitantes en el dominio del árbol, para el mejor entendimiento revisar las siguientes figuras empezando con la **Figura 1.1.**

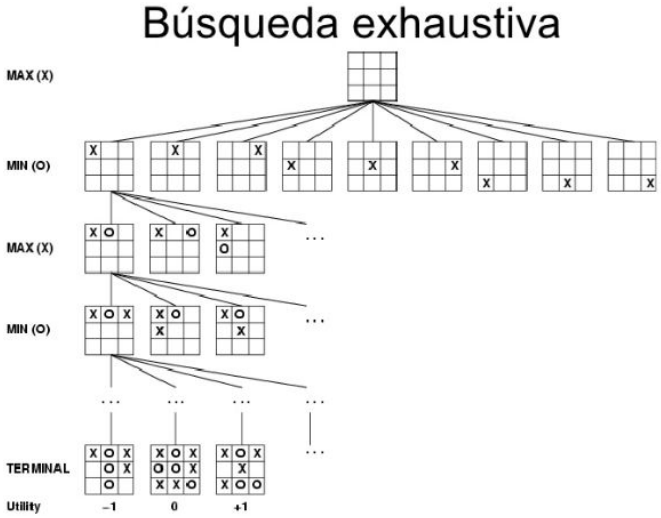


**Figura 1.1**

En la Figura 1.1 se puede observar cómo se genera todo el árbol de posibles jugadas lo cual no es para nada óptimo aunque sigue siendo funcional.

## II. CONTENIDO

Aun para un juego como Tic Tac Toe que es simple y corto, generar todo un árbol completo de las posibles jugadas para realizar la búsqueda exhaustiva con mejor opción. se torna en algo bastante largo y tedioso, para esto se tendrá que aplicar algo llamada aproximación heurística la cual ayudará a optimizar de mejor manera el árbol y las búsquedas que se realizarán dentro de él. En este caso lo que se hará es definir una función que nos indique lo cerca que se está de una jugada ganadora o también de una jugada perdedora, en esta función debe intervenir la información del dominio de la búsqueda, pero sin representar ningún coste ni una distancia en pasos. El algoritmo de esta función debe buscar con una profundidad limitada. Por último cabe resaltar que cada nueva decisión por parte del adversario implica repetir parte de la búsqueda

El algoritmo minimax es una herramienta que ayudará en distintos momentos dentro del juego ya que este se compone de los siguientes acontecimientos. Calcular la decisión minimax del estado actual, usa un cálculo simple recurrente de los valores de minimax de cada estado sucesor, hace que la recursión avance hacia las hojas del árbol y a su vez los valores minimax retroceden por el árbol cuando la recursión se va deshaciendo.

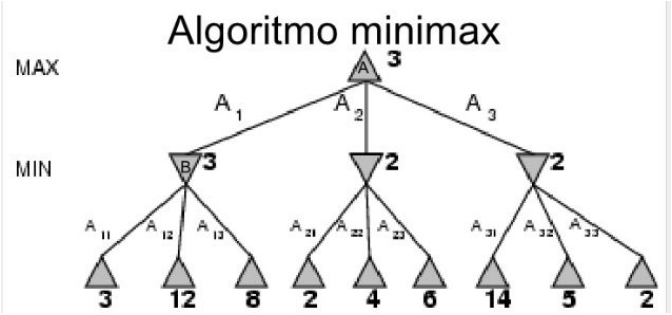


**Figura 2.**

Un ejemplo de cómo funciona este algoritmo es el siguiente. EL algoritmo primero va hacia abajo a los tres nodos izquierdos y utiliza la función Utilidad para descubrir que sus valores son 3, 12 y 8 como se muestra en **Figura 2.**

Entonces el algoritmo toma el mínimo de estos calores que en este caso corresponde al 3 y lo devuelve como el calor del nodo B como se sigue mostrando en la **Figura 2.**

Lo que hace en sí el algoritmo minimax es una exploración en profundidad completa del árbol de juegos. EN el caso de que la profundidad máxima del árbol sea ‘m’ y hay ‘b’ movimientos legales en cada punto, entonces la complejidad es la que se muestra en la **Figura 3**:

- en tiempo es $O(b^m)$;
- en espacio es
  - $O(bm)$ si se generan todos los sucesores a la vez;
  - $O(m)$ si se generan los sucesores uno por uno.

**Figura 3.**

Los juegos reales: los costos de tiempo son inaceptables, pero este algoritmo sirve como base para el primer análisis matemático y para algoritmos más prácticos.

La documentación de las funciones que se deben realizar para el juego Tic Tac Toe se encuentran dentro de la **Figura 4.**

```
función Decisión-Minimax(estado) devuelve una acción
  variables de entrada: estado, estado actual del juego
  v ← Max-Valor(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 ← -∞
  para un s en Sucesores(estado) hacer
    v ← Max(v, Min-Valor(s))
  devolver v

función Min-Valor(estado) devuelve un valor utilidad
  si Test-Terminal(estado) entonces devolver Utilidad (estado)
  v ← ∞
  para un s en Sucesores(estado) hacer
    v ← Min(v, Max-Valor(s))
  devolver v
```

**Figura 4.**

## III. CONCLUSIONES

- Incluso para un juego simple, básico y corto como es el Tic Tac Toe sería demasiado complejo, largo y tedioso realizar un árbol con la totalidad de jugadas posibles.

- Para esta implementación se podría hacer que que una persona jugará contra una inteligencia artificial, pudiendo elegir si quería ser la figura “X” o el “O” o en su defecto poner a jugar a la inteligencia artificial contra ella misma.

- hacer un jugador artificial el cual pueda realizar jugadas intentado simular un ser humano; sin embargo este jugador es demasiado bueno y ganará todas las partidas. En pocas palabras está diseñado para nunca perder.

## RECOMENDACIONES

Ignorar las soluciones redundantes, no hace falta ya que con una vez no es suficiente.

## REFERENCIAS

**[1]** https://es.slideshare.net/LeonardoDaVinciMX/gato-tic-tac toe/72
**[2]** https://es.wikipedia.org/wiki/%C3%81rbol_de_decisi%C 3%B3n
**[3]** https://es.wikipedia.org/wiki/%C3%81rbol_de_decisi%C 3%B3n_(modelo_de_clasificaci%C3%B3n_ID3)
**[4]** https://es.wikipedia.org/wiki/Heur%C3%ADstica

1. Las notas de pie de página deberán estar en la página donde se citan. Letra Times New Roman de 8 puntos