Tic tac toe implementanto algoritmo mimimax en python.

Tic tac toe implementing mimimax algorithm in python.

Autor 1: Jose Daniel Velasquez

*Universidad tecnologica de pereira-Risaralda, Colombia*

Correo-e: j.velasquez@utp.edu.co

***Resumen*— Se implemento mediante el lenguaje de programacion python el juego de tic tac toe. La parte encargada de calcular los movimientos de la maquina se llama Minimax y es un algoritmo que se encarga por medio de busqueda profunda asignar pesos a todos los movimientos para al final indicarle al algoritmo que paso es el indicado para contrarrestar los movimientos del usuario jugador. Es un algoritmo que trabaja principalmente con recursividad.**

***Palabras clave—* Heuristica, minimax, recursividad, busqueda profunda, listas.**

***Abstract*— The tic tac toe game was implemented using the python programming language. The part in charge of calculating the movements of the machine is called Minimax and it is an algorithm that is in charge by means of deep search assigning weights to all the movements to finally indicate to the algorithm which step is the one indicated to counteract the movements of the player user. It is an algorithm that works mainly with recursion.**

***Key Word* —Heuristics, minimax, recursion, deep search, lists.**

1. INTRODUCCIÓN

Tic tac toe, triqui, tres lineas, son los diferentes nombre que puede tener este juego, es un juego de mucha antigüedad y tan conocido como el ajedrez y otros juegos de logica. Consta de un tablero conformado por 9 casillas. Cada casilla puede tomar dos valores, “X” o “O”. A los dos jugadores les corresponde uno de los simbolos (Cabe destacar que los simbolos son los que se utilizan por lo general y en este articulo se utilizaran asi para evitar confuciones, excepto en el algoritmo que se representan como 0 o 1). El objetivo es que alguno de los dos jugadores pueda dibujar 3 simbolos seguidos en tres casillas y asi hacer tres en linea, pueden ser verticales, horizontales y diagonales.

Si se piensa como es la logica detrás del juego, no es dificil ver que según quien empieza el otro tiene que bloquear sus movimientos, como en un juego de atacar y defender. Ahora, si se piensa esto desde la perpectiva computacional, sabemos que no es dificil implementar un juego de estos para dos personas, de hecho seria el caso mas basico de uso del juego,pero si mejor se piensa en un juego de 1vs maquina se pondria mucho mas interesante.

Aquí la pregunta mas importante que se viene a la mente es ¿Cómo hacer para que la maquina juegue, y juegue bien?, ya que en un juego 1 vs 1 no tendriamos que preocuparnos por eso. Lo primero, tal vez, que se puede pensar es en hacer un programa donde todos los movimientos posibles que se pueden hacer esten guardados para luego deacuerdo a un movimiento humano la maquina pueda tener a disposicion todos los movimientos posibles y elegir el mejor. Pero entonces salen aun mas preguntas, ¿como guardar todos estos estados eficientemente?, ¿Qué Heuristica emplear para la toma de desiciones del algoritmo en cuanto al mejor movimietno?, ¿Cómo relizar la implementacion?. A la respuesta de todas estas preguntas se le puede llamar “Minimax”.

1. CONTENIDO

Minimax es un algoritmo de teoria de juegos, se le atribuye principalmente a John Von Neuman, quien probo la teoria fundamental de Minimax en el articulo “Zur Theorie Der Gessellschaftsspiele”(“sobre la teoria de juegos de la sociedad”).

El algoritmo funciona de la siguiente manera, al princpio lo que se hace es definir quien Maximizara y quien Minimizara, despues lo que se hace es generar todos los posibles estados del juego apartir de la forma en que se encuentre el tablero, a estos estados se les asignara un peso, este peso depende de la “funcion heuristica” y de si el estado es MAX o MIN (Maximizador o Minimizador), el estado depende del turno, si es de la maquina o es del jugador.

Imagen que contiene texto, mapa

Descripción generada automáticamente

Figura 1. Sacado de[3]

Para ilustrar mejor como funcona este algoritmo, se puede mirar la Figura 1. Este algoritmo se representa en forma de arbol ya que al hacer la recursividad esta va creando diferentes ramificaciones de diferentes estados. Dependiendo de la ramificacion es Max y Min.

Se puede apreciar algo muy importante y es que los pesos se empizan a generar desde las hojas del arbol, osea desde los estados finales, esto es porque en los estados finales nos damos cuenta si llegamos al objetivo y para eso son los pesos, para guiar el algoritmo hasta el objetivo.

Por ejemplo si nos paramos en el primer nodo, nos damos cuenta que tiene un peso de 7, esto es porque el nodo es de Maximizacion, por lo tanto de sus nodos hijos copia el peso que sea mayor, en este caso el del nodo hijo derecho, y asi recursivamente pasa con el nodo hijo derecho, que copia el peso Minimo de sus hijos, y asi sucesivamente hasta llegar al nodo hoja.

Para este trabajo de investigacion se hizo una implementacion de este algoritmo en Python 3.8.2, la interfaz grafica se desarrolo con la libreria pygame, y se aplico para el juego Tic tac toc.

Imagen que contiene blanco

Descripción generada automáticamente

Figura 2. Interfaz grafica simple del juego, hecha con pygame.

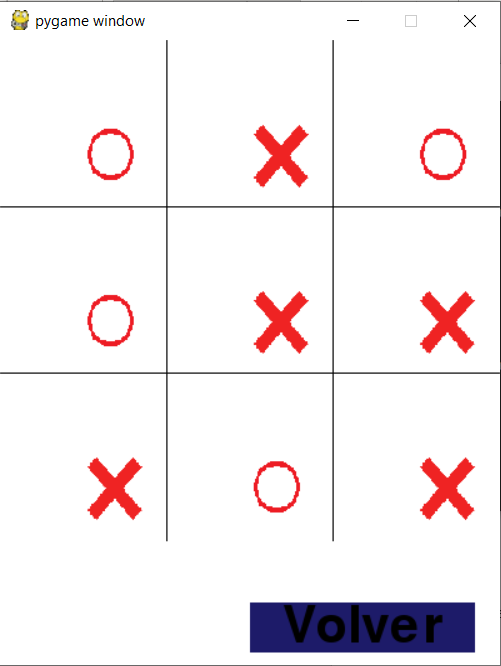


Figura 3. Empate.

En nuestra implementacion:

Jugador 🡪Minimizador y se representa como cero(0) en el tablero.

Maquina 🡪Maximizador y se representa como (1) en el tablero.

La funcion de Heuristica arroja cuatro posibles valores:

0🡪 Para indicar que gano el jugador.

1🡪 Para indicar que gano la maquina.

-100🡪 para el empate

Recordar que estos valores se les da a los nodos hojas y se van replicando de abajo hacia arriba dependiendo de los tipos de nodo, si son MAX o MIN.

Para la representacion del tablero se utilizo una lista de tamaño 9.

Imagen que contiene reloj

Descripción generada automáticamente

Figura 4. Diseño implementacion.

La figura 2 ilustra perectamente la representacion del algoritmo implementado en python y su funcionamiento.

Si analizamos el caso de la imagen nos damos cuenta que al final quien va ganar es la maquina ya que esta se representa por el simbolo “O”.

Algo importante es que el algoritmo hace una busqueda en profundidad por cada movimiento hecho, y hace eso por cada posicion de la lista generando asi el arbol completo

Esta implementacion de tic tac toe, es elegante y rapida, pero un poco completa. Hay una mejora de este algoritmo y se hace llamar “ Poda alfa y beta”, es un algoritmo mejor optimizado ya que va como indica el nombre podando ramificaciones que no sirven. Para este articulo no se tuvo encuenta este tipo de algoritmo y solo se centro en el Minimax original.

La implementacion estara disponible en el repositorio de github.

1. CONCLUSIONES

* Algo importante a analizar es la complejidad que esconde un simple juego como lo es tic tac toe, para los humanos puede ser muy facil, pero pasar al mundo computacional y tratar de modelar el comportamiento humano puede ser de gran complejidad.
* El algoritmo trabaja como una estructura de datos de arboles binarios, por lo tanto su tiempo de ejecucion no es muy alto.
* Si se juega muchas veces se puede analisar que con la implementacion de este algoritmo es imposible ganarle a la maquina ya que siempre va tomar las mejores decisiones y estas al final llevan al empate, por lo tanto en el peor de los casos se puede perder con la maquina y en el mejor empatar. Esto demuestra la calidad del algoritmo.

REFERENCIAS

1. Articulo “Triqui – AlgoritmoMinimax” de https://sites.google.com/site/inteligenciaartificial022014/triqui
2. Articulo “Solving Tic-Tac-Toe, Part II: A Better Way” de https://catarak.github.io/blog/2015/01/07/solving-tic-tac-toe/
3. Articulo “MiniMax Algorithm in Game Theory and C implementation” de <https://csit-notes.blogspot.com/2019/08/minimax-algorithm-in-game-theory-and-c.html>
4. IA – Algoritmos de Juegos de Pau Aguilar Fruto, Universidad politecnica de cataluña.