UNIVERSIDAD SIMÓN BOLÍVAR

DEPARTAMENTO DE COMPUTACIÓN Y TECNOLOGÍA DE LA INFORMACIÓN

CI5437 – INTELIGENCIA ARTIFICIAL I

Trimestre: Enero – Marzo 2024

Profesor: Carlos Infante

Informe del Proyecto #2

Implementación de la Algoritmos de Árboles para el Juego Othello 6x6

Estudiantes:

José Matías González, 15-10627

Gabriel Chaurio, 17-10126

Ana Santos, 17-10602

Sartenejas, Marzo del 2023

ÍNDICE

DEFINICIÓN DEL PROBLEMA

El juego Othello 6x6 es un juego de mesa comienza con cuatros (4) fichas en le centro, dos (2) blancas y dos (2) negras. Los jugadores se turnan para colocar una ficha de su color en el tablero de tal manera que se busca encerrar una o más fichas del oponente entre la ficha que se coloca y cualquier otra dicha de su color ya en el tablero. Todas las fichas del oponente que quedan encerradas deben cambiar de color al del jugador que haya realizado la jugada. El juego continúa hasta que el tablero está lleno o ninguno de los jugadores puede hacer una jugada válida. El ganador es el jugador con más fichas de su color en el tablero al final del juego.

Para el curso de Inteligencia Artificial I, el juego Othello puede ser modelado como un problema de búsqueda del árbol de juego, donde cada nodo del árbol representa un estado de juego y cada arista representa una jugada válida. Los algoritmos de búsqueda en árboles, como los definidos a continuación, pueden ser utilizados para explorar este árbol de juego y determinar la mejor jugada en cada turno.

Por ello, se busca que los estudiantes de este curso adquieran un entendimiento sólido de cómo se modelan problemas de este tipo en Inteligencia Artificial, utilizando árboles de juego y cómo se pueden resolver utilizando algoritmos de búsqueda en árboles. Los algoritmos a estudiar son Negamax sin poda alfa-beta, Negamax con poda alfa-besta, Scout y Negascout, de los cuales se querrá medir la eficiencia. Finalmente, se quiere la presentación de un análisis de los resultados obtenidos.

METODOLOGÍA

Los algoritmos de búsqueda de árboles deben que se usaron en este proyecto fueron completados en el lenguaje C y mientras que para el análisis de resultados se realizó en Python para poder visualizar las gráficas.

1. Implementación de los Algoritmos de Árboles

a. Negamax sin poda alfa-beta (versión minimax)

Este procedimiento es una variante del algoritmo Minimax que se basa en la propiedad de suma cero de un juego de dos (2) jugadores. Este algoritmo simplifica la implementación del Minimax al asumir que el valor de una posición para un jugador es la negación del valor para el otro jugador.

b. Negamax con poda alfa-beta

Este algoritmo es una mejora del anterior que incorpora la técnica de poda alfa-beta para reducir el número de nodos que se evalúan en el árbol de búsqueda. La poda alfa-beta permite evitar la búsqueda de ciertas posiciones que no influirán en la decisión final.

c. Scout

Este algoritmo es una variante de la búsqueda en árboles de juefo que se utiliza para encontrar y recuperar información de manera rápida y eficiente. En el contexto de los juegos, el algoritmo Scout se utiliza para realizar una búsqueda de la variante principal del juego, es decir, para explorar el árbol de juego en busca de la mejor jugada sin tener que examinar todas las posibles variantes del juego.

d. Negascout

Este algoritmo es una variante del algoritmo Minimax que combina la relación matemática del Negamax y el uso de ventana nula de Scout. Puede ofrecer rendimientos de poda incluso mayores que alfa-beta si los nodos se encuentran correctamente ordenados.

2. Especificaciones del Equipo usado

Todas las pruebas se corrieron en el mismo equipo, el cual cuenta con las siguientes especificaciones:

- Procesador: Intel i5-12500H, 12 Cores.
- RAM: 64GB, 3100MHz.
- SSD

RESULTADOS Y ANÁLISIS

1. Nodos expandidos durante las búsquedas

El Gráfico 1 muestra el comportamiento asintótico de los nodos expandidos por cada nivel PV para los cuatro algoritmos de árboles de juego con los algoritmos antes escritos. Todos los algoritmos denotan un comportamiento esperado ya que a medida que se profundiza en el árbol de juego, el número de posibles estados del juego (nodos) aumenta.

Sin embargo, hay diferencias significativas en la tasa de crecimiento entre los algoritmos. Algunos algoritmos, como Negamax, muestran un crecimiento más rápido, lo que indica que expanden más rápido, lo que indica que expanden un mayor número de nodos. Otro algoritmos, como Negascout, muestran un crecimiento más lento, lo que indica que son más eficientes en la expansión de nodos.

Estas diferencias en el crecimiento pueden atribuirse a las técnicas utilizadas por cada algoritmo para explorar el árbol de juegos. Por ejemplo, Negamax Alfa-Beta y Negascout, utilizan técnicas de poda para reducir el número de nodos que necesitan ser expandidos, mientras que, Scout utilizan estrategias de búsqueda específicas para explorar el árbol de juego de manera más eficiente.

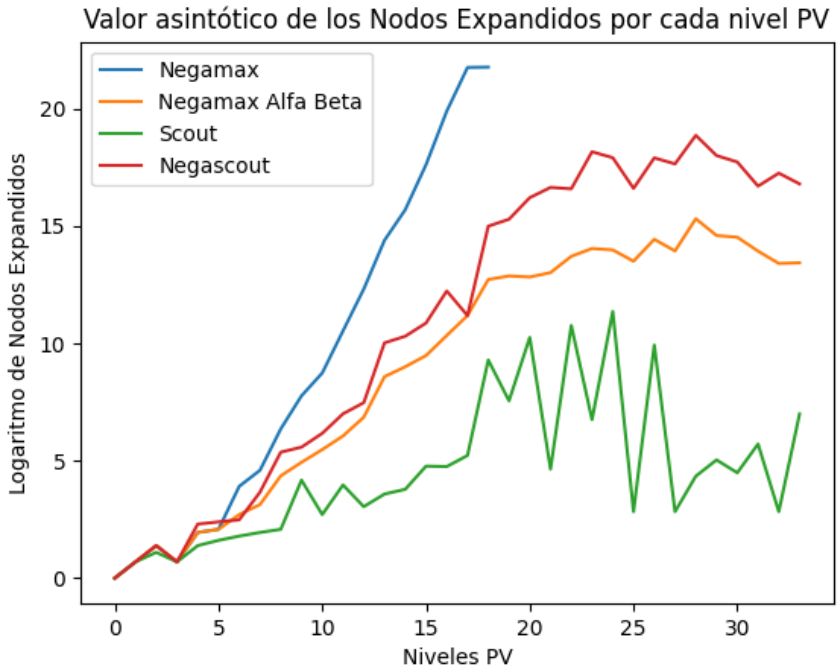

Gráfico 1. *Valor asintótico de los Nodos expandidos por cada nivel PV*

2. Nodos generados durante las búsquedas

En el Gráfico 2, todos los algoritmos muestran un aumento en el número de nodos generados a medida que aumenten los niveles PV. Se observa que Negamax sin poda genera una cantidad significativamente mayor de nodos en comparación con los otros métodos. A medida que aumentan los niveles PV, la cantidad de nodos generados por este algoritmo también aumenta drásticamente. Por otro lado, Negamax con poda Alfa-beta, Scout, Negascout muestran un crecimiento más moderado a medida que aumentan los niveles PV.

Es importante tener en cuenta que menos nodos generados significa una búsqueda más eficiente. Por lo tanto, Negamax con poda, Scout y Negascout parecer ser más eficientes que Negamax en este apartado.

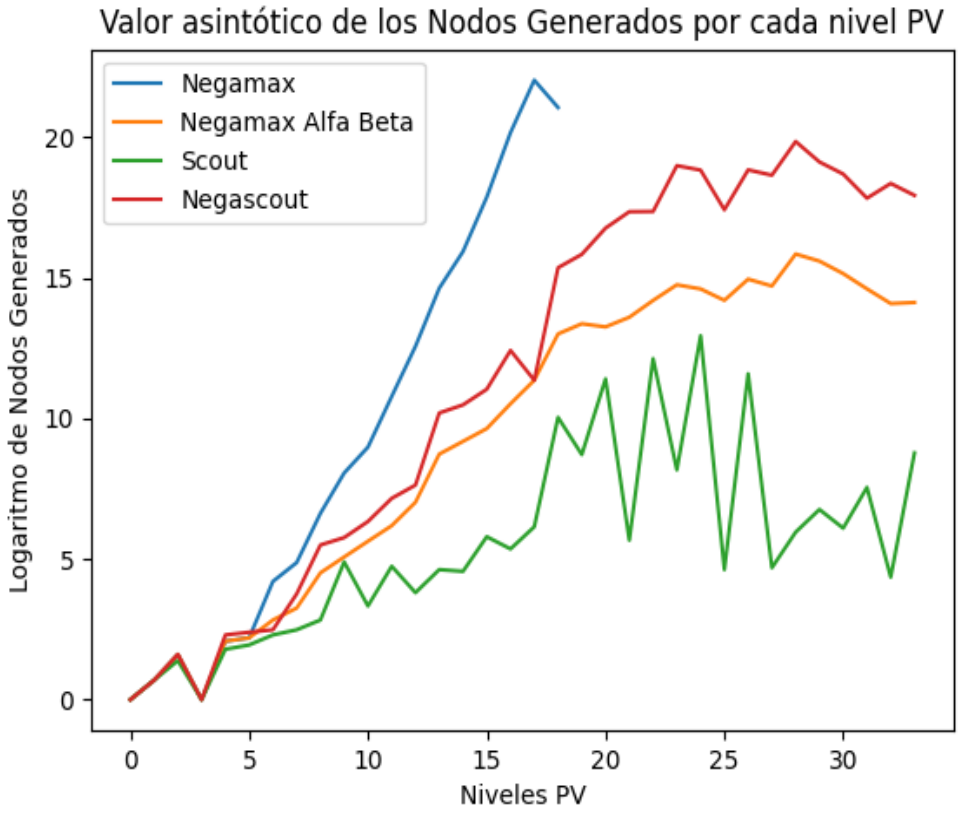

Gráfico 2. *Valor asintótico de los Nodos generados por cada nivel PV*

3. Tiempo en segundos

En el análisis del valor asintótico del tiempo por cada nivel, se encontró que Negamax muestra un aumento significativo a medida que aumentan los niveles PV. Específicamente, después del nivel PV 15, el tiempo asintótico aumenta de manera exponencial. Por otro lado, los otros tres (3) algoritmos mantienen un tiempo asintótico bajo y constante a lo largo de los niveles. Esto sugiere que son más eficientes en términos de tiempo en comparación con Negamax.

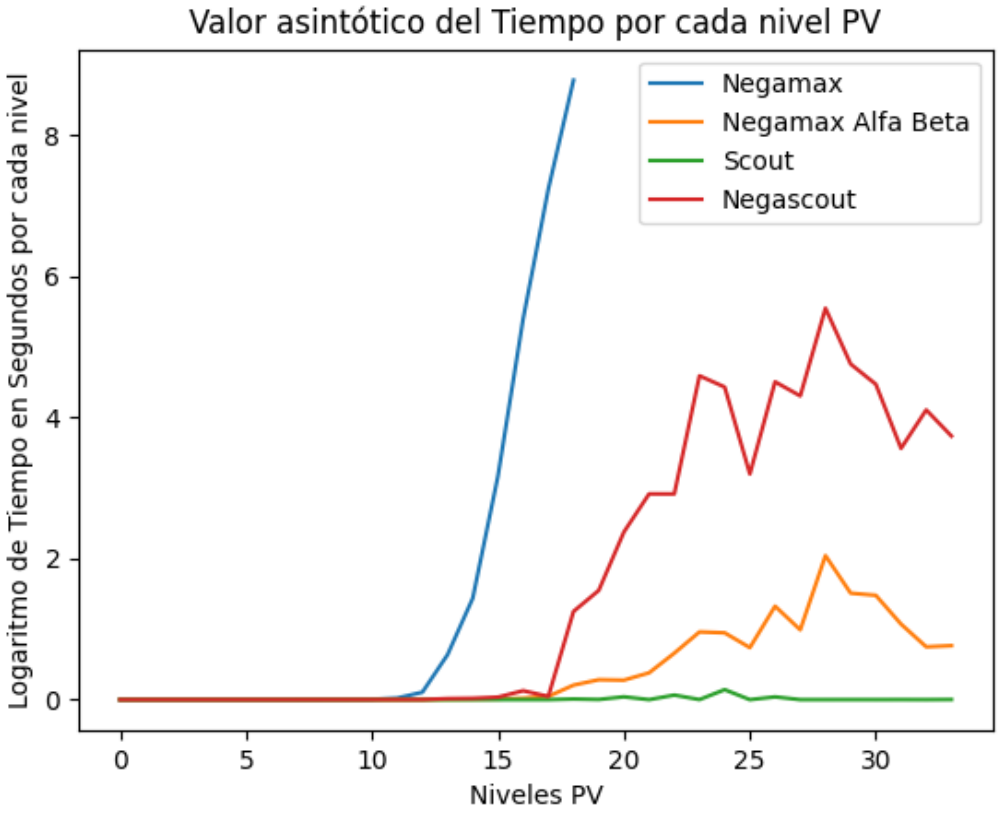

Gráfico 3. *Valor asintótico del tiempo de corrida de cada nivel PV.*

4. Nodos generados por segundos durante las búsquedas

Todos los algoritmos muestran un comportamiento similar en términos de nodos generados por segundo a lo largo de los niveles PV. Después de un pico inicial, la eficiencia parece ser constante y no muestra variaciones significativas entre los diferentes algoritmos.

Por lo que se concluye, que en términos de nodos generados Negamax genera significativamente más nodos que los otros algoritmos, por lo que, Negamax también requiere más tiempo que los otros. Sin embarho en términos de nodos generados todos los algoritmos muestran un comportamiento similar. Por lo tanto, los tres (3) últimos algoritmos podrían ser preferibles para el caso de Othello 6x6.

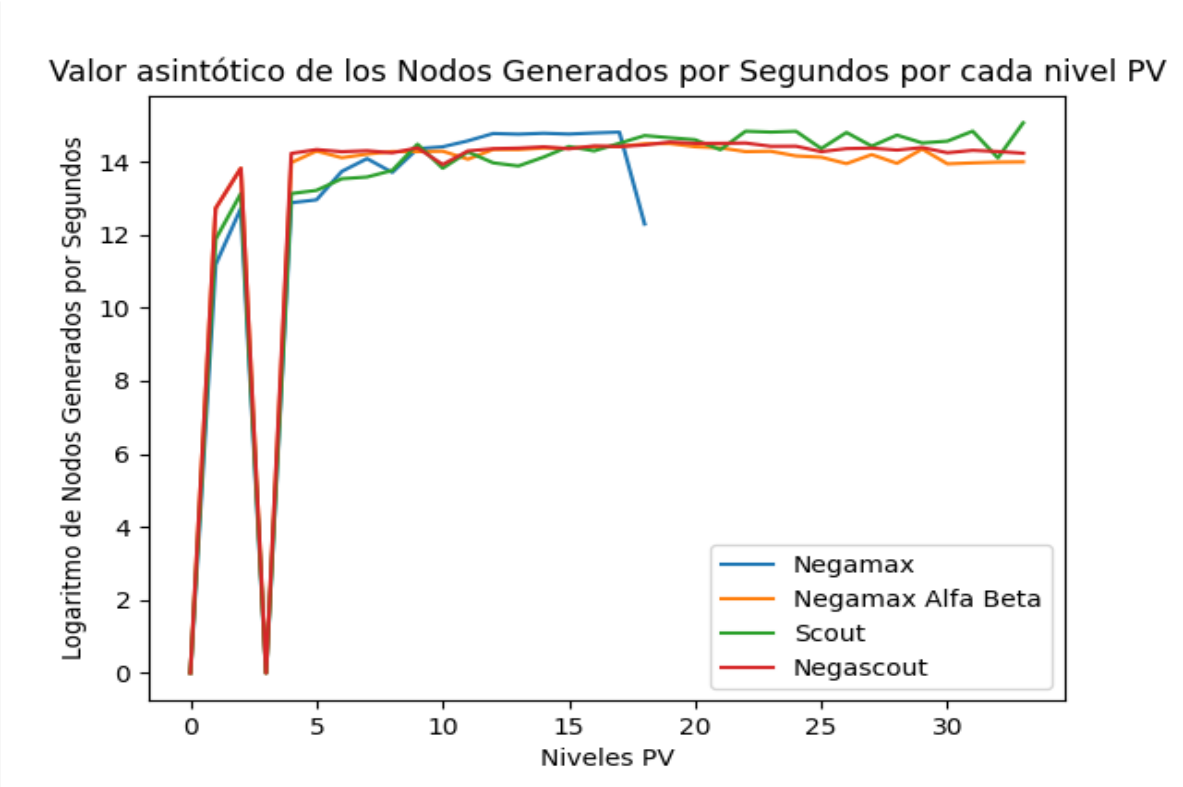

Gráfico 4. *Valor asintótico de los Nodos Generados por Segundos por cada nivel PV*