

Santiago, 20 de septiembre de 2021

Reversi: aplicando MiniMax

Integrantes:

Gabriel Silva Seguel
Guillermo Ahumada Logan
Lucy Ramos Silva
Benjamín Vilches Cuadra

Profesor: Pablo Schwarzenberg Riveros

Profesor Ayudante: Javier Salazar Loyola

GitHub: <https://github.com/Bemhaha/Reversi-minimax>



Índice

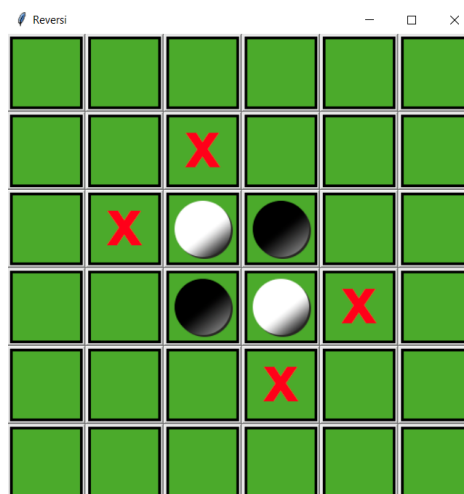
1. Introducción al contexto del problema.....	2
2. Estrategias de juego y justificación.....	4
3. Representación y jugadas.....	6
4. Función de utilidad.....	7
5. Dificultad del juego.....	10
6. Conclusiones respecto al desempeño y ejemplos de juegos.....	11



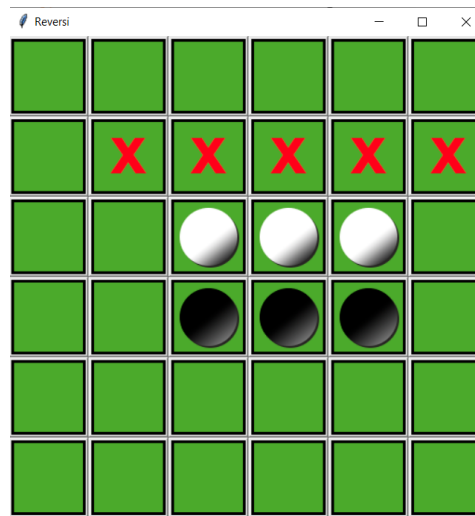
1. Introducción al contexto del problema.

Reversi u Othello es un juego de dos jugadores, que comúnmente se desarrolla en un tablero de 8x8, con fichas de color negro o blanco. En este caso, nuestro desarrollo se basó en un tablero de 6x6.

Al iniciar el juego se posicionan 4 fichas en el centro del juego, 2 blancas y 2 negras, posicionadas en diagonal cada color.



El objetivo del jugador es colocar en el tablero una ficha de su color que rodee una o más fichas de su oponente para voltearlas y hacerlas suyas, a lo que llamamos hacer “reversi” a las fichas. Para poder ganar este juego, debemos lograr acumular la mayor cantidad de fichas de nuestro color en el tablero, las jugadas válidas se logran cuando tu ficha logra “saltar a la otra”, en la imagen anterior marcadas con una X de color rojo.

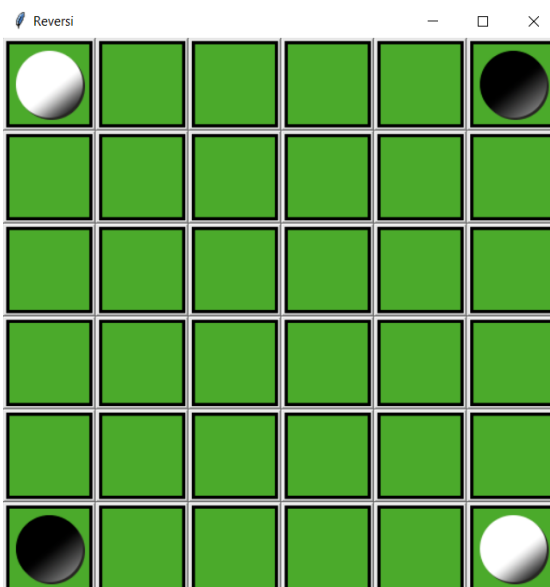


Una vez colocada tu ficha, se darán vuelta a las otras para así obtener más fichas de tu color, para luego poder llevarse la victoria del juego. En la imagen anterior podemos ver cómo actúa el “hacer reversi” a las fichas según como fueron puestas y las nuevas jugadas posibles.

2. Estrategias de juego y justificación.

La estrategia más simple que se podría pensar en reversi es elegir las jugadas que te permitan conseguir dar vuelta o hacer reversi a la mayor cantidad de fichas posibles, pero dada la evidencia que se encuentra en los diversos juegos que se pueden observar, se puede concluir que esta estrategia es muy poco eficiente, ya que puede llevar de manera muy fácil a perder.

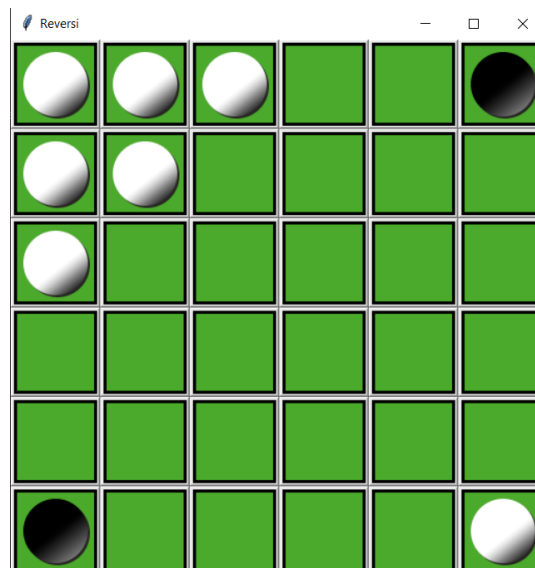
Una de las estrategias usadas en este juego fue la estrategia “mejor esquina”, con la cual logramos ver que al conseguir tener nuestra ficha en una esquina se incrementan mucho las posibilidades de ganar, dado que en esta posición el jugador enemigo no puede “hacer reversi”, por lo cual será una ficha que siempre estará con el mismo color y el rival no podrá hacer nada. Para nuestra representación cuando se lograba esta jugada, se le asignaban 25 puntos al tablero.



Otro de los fuertes de esta estrategia es que dado que logramos posicionar nuestra ficha en una esquina, esto nos ayuda a poder atrapar de manera mucho más simple las fichas enemigas y así hacerles “Reversi” a sus fichas. El único problema de esta jugada, es que para lograrla debemos realizar unas cuantas jugadas antes, lo cuál le da chance al jugador enemigo de poder contrarrestar nuestra estrategia.



La otra estrategia utilizada fue la de elegir los laterales y adyacentes a las esquinas, estas jugadas nos dan un beneficio un poco mayor, pero es muy complicada de lograr, dado que requerimos de muchas jugadas para llegar a esta.



Por otro lado, estas fichas laterales o adyacentes no pueden ser capturadas, pero como ya dijimos, dada su complejidad, el puntaje asignado es de 5 puntos una vez que se logra.



3. Representación y jugadas.

La representación escogida fue la de una matriz de 6x6, consideramos que esta era la mejor forma para que el usuario entendiera lo que ocurre en cada turno y pueda jugar de manera acorde.

Decidimos que esta forma era mejor que otras al ser más simple su uso, puesto que una matriz de 6x6 simula el tablero del juego. A diferencia de utilizar un arreglo muy grande, listas de adyacencia o una variable por celda, nuestro método nos permite una visión más clara de lo que ocurre, logrando que sea más fácil resolver los problemas encontrados y llegar a un código óptimo, además de facilitar mucho las trazas hechas a mano y poder visualizar mentalmente como programador lo que está pasando dentro de la computadora.

Aij	0	1	2	3	4	5
0	"0"	"0"	"0"	"0"	"0"	"0"
1	"0"	"0"	"0"	"0"	"0"	"0"
2	"0"	"0"	"-1"	"1"	"0"	"0"
3	"0"	"0"	"1"	"-1"	"0"	"0"
4	"0"	"0"	"0"	"0"	"0"	"0"
5	"0"	"0"	"0"	"0"	"0"	"0"

En síntesis, nuestra elección se basó en la representación de una matriz dado su fácil uso y fácil entendimiento por parte del programador. Como se puede ver en la imagen, identificamos como "-1" a las fichas blancas, "1" a las fichas negras y "0" a los cuadros vacíos.



4. Función de utilidad

Para poder desarrollar minimax utilizamos un pseudo código estándar, el cuál se basa en crear el árbol de juego, en base a una función recursiva, teniendo una condición base, una para Max o derecha de árbol y otra para Min o izquierda del árbol.

```
function minimax(node, depth, maximizingPlayer) is
  if depth = 0 or node is a terminal node then
    return the heuristic value of node
  if maximizingPlayer then
    value := -∞
    for each child of node do
      value := max(value, minimax(child, depth - 1, FALSE))
    return value
  else (* minimizing player *)
    value := +∞
    for each child of node do
      value := min(value, minimax(child, depth - 1, TRUE))
    return value
```

Para la condición base, donde se cumple que llegamos al final del árbol con un posible tablero futuro (**nuestro juego va creando futuros tableros posibles según se va llamando recursivamente a MiniMax**), debemos evaluar este tablero dándole un puntaje, es por esto que acudimos a una función de utilidad, la cuál nos dirá qué puntaje tiene el tablero del nodo terminal según una serie de estrategias de juego que se nombran en el punto 2.

```
1  def heuristicaMejorEsquina(self, tablero, ficha):
2
3      puntajeHeuristica = 0
4      puntajeEsquina = 25
5      puntajeAdyacente = 5
6      puntajeLateral = 5
7
8      if ficha == 1:
9          ficha = "1"
10         fichaEnemiga = "-1"
11     else:
12         ficha = "-1"
13         fichaEnemiga = "1"
```

En primer lugar lo que hacemos es dar puntajes a las jugadas, iniciamos la suma total o "puntajeHeuristica" en 0, luego le damos el valor de 25 puntos si encontramos una esquina, 5 si es una ficha lateral adyacente y 5 si es lateral, además vemos cuál



es la ficha aliada y cual es la enemiga, una vez evaluado esto, podemos comenzar a “dar un puntaje al nodo o tablero futuro jugado por minimax”.

```
for x in range(6):
    for y in range(6):
        sumaAux = 1
        if (x == 0 and y == 1) or (x == 1 and 0 <= y <= 1):
            if tablero[0][0] == ficha:
                sumaAux = puntajeLateral
            else:
                sumaAux = -puntajeAdyacente

        elif (x == 0 and y == 4) or (x == 1 and 4 <= y <= 5):
            if tablero[5][0] == ficha:
                sumaAux = puntajeLateral
            else:
                sumaAux = -puntajeAdyacente
        elif (x == 5 and y == 1) or (x == 4 and 0 <= y <= 1):
            if tablero[0][5] == ficha:
                sumaAux = puntajeLateral
            else:
                sumaAux = -puntajeAdyacente
        elif (x == 5 and y == 4) or (x == 4 and 4 <= y <= 5):
            if tablero[5][5] == ficha:
                sumaAux = puntajeLateral
            else:
                sumaAux = -puntajeAdyacente
        elif (x == 0 and 1 < y < 4) or (x == 5 and 1 < y < 4) or (y == 0 and 1 < x < 4) or (y == 5 and 1 < x < 4):
            sumaAux = puntajeLateral
        elif (x == 0 and y == 0) or (x == 0 and y == 5) or (x == 5 and y == 0) or (x == 5 and y == 5):
            sumaAux = puntajeEsquina
```

La función de este doble ciclo es de revisar los adyacentes, los laterales y las esquinas, evaluar si la ficha posicionada en ese lugar es enemiga o aliada, para poder dar un puntaje al tablero, y así revisa cada celda de la matriz 6x6, cabe destacar como se nombró anteriormente, que nuestros nodos en minmax, no son tuplas (x,y) de una jugada, sino la matriz completa de 6x6, con una jugada posible realizada por minimax, para profundizar cómo funciona este algoritmo, este se encuentra completamente comentado en el repositorio <https://github.com/Bemhaha/Reversi-minimax.git>

```
if tablero[x][y] == ficha:
    puntajeHeuristica += sumaAux
elif tablero[x][y] == fichaEnemiga:
    puntajeHeuristica -= sumaAux

return puntajeHeuristica
```



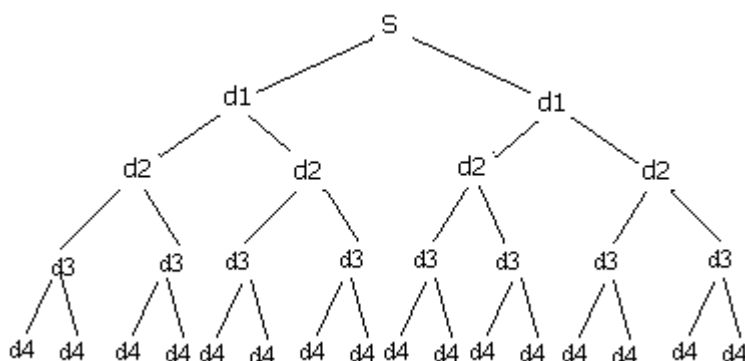
Para terminar, esta sentencia if elif, se encuentra al final del doble ciclo “for”, y lo que hace es ver si en la celda que se encuentra evaluando hay una ficha aliada o enemiga, dependiendo de que sea, sumará o restará el puntaje evaluado.

Se termina esta función de utilidad retornando el puntaje evaluado del nodo o tablero posible futuro, para que minimax siga buscando en otros nodos y pueda elegir la jugada óptima.



5. Dificultad del juego

La solución que encontramos para la dificultad fue una muy sencilla y eficaz, simplemente hicimos que el árbol de búsqueda de MiniMax fuese más pequeño para una dificultad más fácil, en este caso, una profundidad de búsqueda de 4 para el modo “Difícil”, 3 para el “Normal” y 2 para el “Fácil”. Al hacer esto la computadora actúa de formas diferentes, dado que manejamos las posibles jugadas futuras que verá y evaluará minimax, así logramos generar los diferentes niveles de dificultad para el jugador humano.

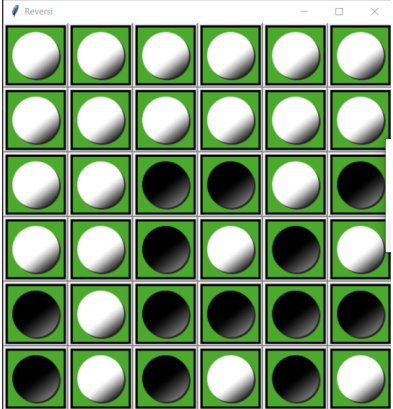


En la imagen anterior se puede ver como va creciendo un árbol de búsqueda y cómo revisa más opciones a medida que más profundidad o niveles tiene, lo cuál le da al final la diferencia de dificultad de cada modo de juego, cabe destacar que es una imagen referencial, el minimax que aplicamos en Reversi genera una cantidad mucho mayor de nodos por nivel.

A diferencia de otras formas de cambio de dificultad, ya sea una heurística menos sofisticada, o un algoritmo que juegue aleatoriamente, nuestro método de selección de dificultad logra llegar a simular el cómo jugaría un humano con menos o más experiencia, basándose en ideas o heurísticas de juego conocidas pero no tan desarrolladas, dependiendo de la profundidad de la búsqueda.

6. Conclusiones con respecto al desempeño y ejemplos de juegos.

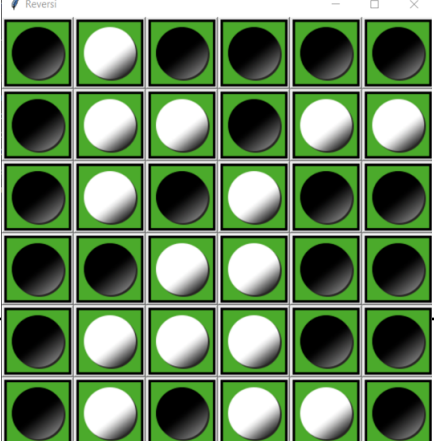
Antes de poder hacer una análisis general y sacar conclusiones generales, vamos a mostrar los resultados de prueba con los distintos niveles de dificultad que aplicamos para el juego Reversi.



DIFICULTAD = FÁCIL
PROFUNDIDAD DE BUSQUEDA = 2
RESULTADO JUEGO = GANA BLANCO / GANA COMPUTADORA
CANTIDAD NODOS Y TIEMPO EJECUCIÓN POR LLAMADA =

La cantidad de nodos explorados por minimax fueron = 13
El tiempo que tardó minimax en explorar los nos fue de = 0.0059812068939208984 segundos
La cantidad de nodos explorados por minimax fueron = 43
El tiempo que tardó minimax en explorar los nos fue de = 0.012010812759399414 segundos
La cantidad de nodos explorados por minimax fueron = 73
El tiempo que tardó minimax en explorar los nos fue de = 0.020953655242919922 segundos
La cantidad de nodos explorados por minimax fueron = 7
El tiempo que tardó minimax en explorar los nos fue de = 0.0019538402557373047 segundos
La cantidad de nodos explorados por minimax fueron = 31
El tiempo que tardó minimax en explorar los nos fue de = 0.008019208908081055 segundos
La cantidad de nodos explorados por minimax fueron = 43
El tiempo que tardó minimax en explorar los nos fue de = 0.009014606475830078 segundos
La cantidad de nodos explorados por minimax fueron = 43
El tiempo que tardó minimax en explorar los nos fue de = 0.011010169982910156 segundos
La cantidad de nodos explorados por minimax fueron = 57
El tiempo que tardó minimax en explorar los nos fue de = 0.0139617919921875 segundos
La cantidad de nodos explorados por minimax fueron = 73
El tiempo que tardó minimax en explorar los nos fue de = 0.023972272872924805 segundos
La cantidad de nodos explorados por minimax fueron = 57
El tiempo que tardó minimax en explorar los nos fue de = 0.0149993896484375 segundos
La cantidad de nodos explorados por minimax fueron = 43
El tiempo que tardó minimax en explorar los nos fue de = 0.008971452713012695 segundos
La cantidad de nodos explorados por minimax fueron = 43
El tiempo que tardó minimax en explorar los nos fue de = 0.011970043182373047 segundos
La cantidad de nodos explorados por minimax fueron = 31
El tiempo que tardó minimax en explorar los nos fue de = 0.009009122848510742 segundos
La cantidad de nodos explorados por minimax fueron = 31
El tiempo que tardó minimax en explorar los nos fue de = 0.013921260833740234 segundos
La cantidad de nodos explorados por minimax fueron = 13
El tiempo que tardó minimax en explorar los nos fue de = 0.0010132789611816406 segundos
La cantidad de nodos explorados por minimax fueron = 3
El tiempo que tardó minimax en explorar los nos fue de = 0.0 segundos

En el primer nivel de dificultad, es decir, “fácil”, con profundidad de búsqueda 2 podemos ver lo rápido que actúa minimax, y la poca cantidad de nodos que explora, en ningún caso se llegó a tardar en buscar más de 0.1 segundos, ni siquiera en “mid game” que es cuando más nodos se exploran.



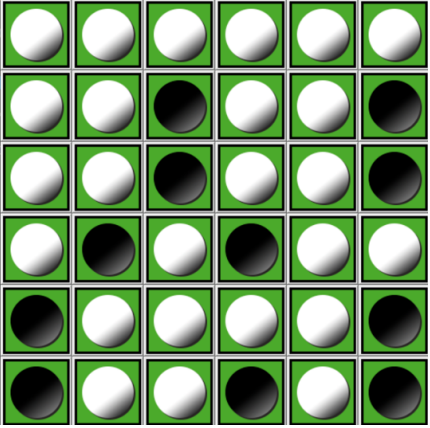
DIFICULTAD = NORMAL
PROFUNDIDAD DE BUSQUEDA = 3
RESULTADO JUEGO = GANA NEGRO / GANA USUARIO
CANTIDAD NODOS Y TIEMPO EJECUCIÓN POR LLAMADA =

La cantidad de nodos explorados por minimax fueron = 40
El tiempo que tardó minimax en explorar los nos fue de = 0.012966156005859375 segundos
La cantidad de nodos explorados por minimax fueron = 259
El tiempo que tardó minimax en explorar los nos fue de = 0.08274650573730469 segundos
La cantidad de nodos explorados por minimax fueron = 85
El tiempo que tardó minimax en explorar los nos fue de = 0.025959014892578125 segundos
La cantidad de nodos explorados por minimax fueron = 259
El tiempo que tardó minimax en explorar los nos fue de = 0.06822347640991211 segundos
La cantidad de nodos explorados por minimax fueron = 820
El tiempo que tardó minimax en explorar los nos fue de = 0.27430033683776855 segundos
La cantidad de nodos explorados por minimax fueron = 1111



	<p>El tiempo que tardó minimax en explorar los nos fue de = 0.38475894927978516 segundos La cantidad de nodos explorados por minimax fueron = 820 El tiempo que tardó minimax en explorar los nos fue de = 0.26030802726745605 segundos La cantidad de nodos explorados por minimax fueron = 156 El tiempo que tardó minimax en explorar los nos fue de = 0.03391456604003906 segundos La cantidad de nodos explorados por minimax fueron = 400 El tiempo que tardó minimax en explorar los nos fue de = 0.10571980476379395 segundos La cantidad de nodos explorados por minimax fueron = 400 El tiempo que tardó minimax en explorar los nos fue de = 0.1047210693359375 segundos La cantidad de nodos explorados por minimax fueron = 259 El tiempo que tardó minimax en explorar los nos fue de = 0.05788064002990723 segundos La cantidad de nodos explorados por minimax fueron = 259 El tiempo que tardó minimax en explorar los nos fue de = 0.05817103385925293 segundos La cantidad de nodos explorados por minimax fueron = 40 El tiempo que tardó minimax en explorar los nos fue de = 0.005026102066040039 segundos La cantidad de nodos explorados por minimax fueron = 85 El tiempo que tardó minimax en explorar los nos fue de = 0.016949892044067383 segundos La cantidad de nodos explorados por minimax fueron = 40 El tiempo que tardó minimax en explorar los nos fue de = 0.003983974456787109 segundos La cantidad de nodos explorados por minimax fueron = 4 El tiempo que tardó minimax en explorar los nos fue de = 0.0 segundos</p>
--	---

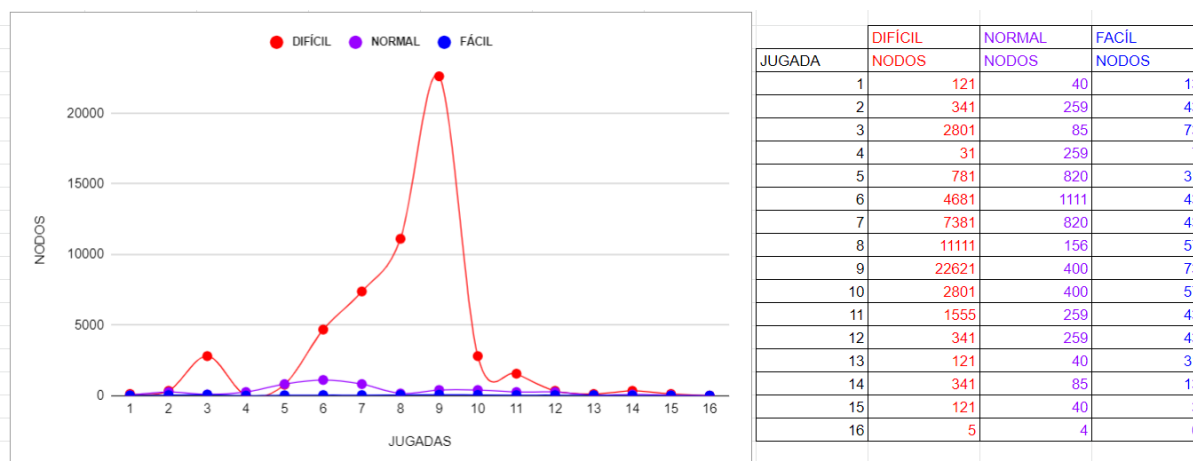
En el segundo nivel de dificultad, es decir “normal”, con profundidad de búsqueda 3, podemos ver que Minimax tarda un poco más en buscar la jugada óptima, llegando a demorar tan solo 0.3 segundos en su momento más tardío, el usuario logra sentir que son jugadas con un nivel de estrategia más alto, similar a como sería con un jugador humano promedio, al igual que el caso de prueba anterior, en mid game es cuando más nodos se exploran y más tarda en buscar.

	<p>DIFICULTAD = DIFÍCIL PROFUNDIDAD DE BUSQUEDA = 4 RESULTADO JUEGO = GANA BLANCO / GANA COMPUTADORA CANTIDAD NODOS Y TIEMPO EJECUCIÓN POR LLAMADA =</p> <p>La cantidad de nodos explorados por minimax fueron = 121 El tiempo que tardó minimax en explorar los nos fue de = 0.025971174240112305 segundos La cantidad de nodos explorados por minimax fueron = 341 El tiempo que tardó minimax en explorar los nos fue de = 0.07475972175598145 segundos La cantidad de nodos explorados por minimax fueron = 2801 El tiempo que tardó minimax en explorar los nos fue de = 0.8179998397827148 segundos La cantidad de nodos explorados por minimax fueron = 31 El tiempo que tardó minimax en explorar los nos fue de = 0.008956670761108398 segundos La cantidad de nodos explorados por minimax fueron = 781 El tiempo que tardó minimax en explorar los nos fue de = 0.18251323699951172 segundos La cantidad de nodos explorados por minimax fueron = 4681 El tiempo que tardó minimax en explorar los nos fue de = 1.3624029159545898 segundos La cantidad de nodos explorados por minimax fueron = 7381 El tiempo que tardó minimax en explorar los nos fue de = 2.358638286590576 segundos La cantidad de nodos explorados por minimax fueron = 11111 El tiempo que tardó minimax en explorar los nos fue de = 3.604764461517334 segundos La cantidad de nodos explorados por minimax fueron = 22621 El tiempo que tardó minimax en explorar los nos fue de = 8.002513647079468 segundos La cantidad de nodos explorados por minimax fueron = 2801 El tiempo que tardó minimax en explorar los nos fue de = 0.671391487121582 segundos La cantidad de nodos explorados por minimax fueron = 1555 El tiempo que tardó minimax en explorar los nos fue de = 0.3148374557495117 segundos La cantidad de nodos explorados por minimax fueron = 341 El tiempo que tardó minimax en explorar los nos fue de = 0.05285072326660156 segundos La cantidad de nodos explorados por minimax fueron = 121 El tiempo que tardó minimax en explorar los nos fue de = 0.021981000900268555 segundos La cantidad de nodos explorados por minimax fueron = 341 El tiempo que tardó minimax en explorar los nos fue de = 0.04590797424316406 segundos La cantidad de nodos explorados por minimax fueron = 121 El tiempo que tardó minimax en explorar los nos fue de = 0.013004064559936523 segundos La cantidad de nodos explorados por minimax fueron = 5 El tiempo que tardó minimax en explorar los nos fue de = 0.0009672641754150391 segundos</p>
---	--

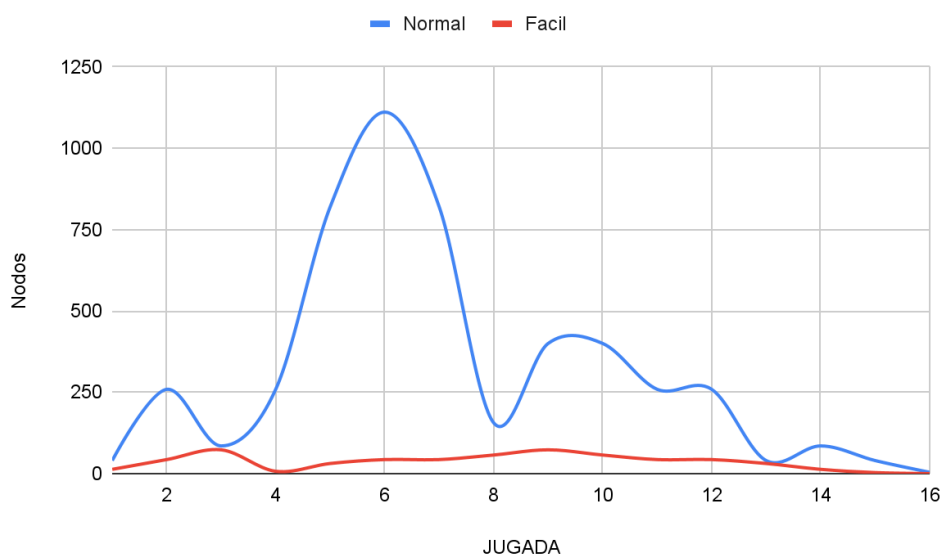
En el último modo de juego, es decir “difícil”, con profundidad de búsqueda 4, podemos ver como minimax explora una cantidad mucho más grandes de nodos, llegando en mid game a explorar 22.000 nodos, incluso en algunas pruebas se llegó



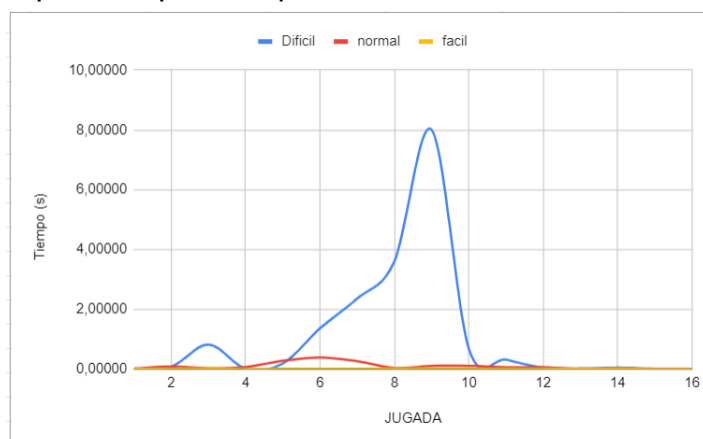
a tener 50.000 nodos, cuando llegamos a este nivel, el tiempo que demora minimax en encontrar la jugada óptima, puede variar entre 8 segundos como en este caso, hasta incluso un tope de 30 segundos, en esta situación se acerca mucho más a lo que sería una persona, pero de todas formas se llega a sentir algo tardío.



Como se puede apreciar existe una tendencia constante en cada uno de los gráficos, formando una distribución normal o de Gauss (similar a una campana), básicamente significa que la cantidad de nodos explorados es considerablemente menor al inicio y al final de cada partida en comparación al juego medio (*Mid game*), tras analizar este comportamiento se ha determinado que existen dos factores principales de los cuales depende la extensión de cada exploración, estos factores se definen como: Cantidad de fichas en tablero y Cantidad de celdas vacías del tablero.

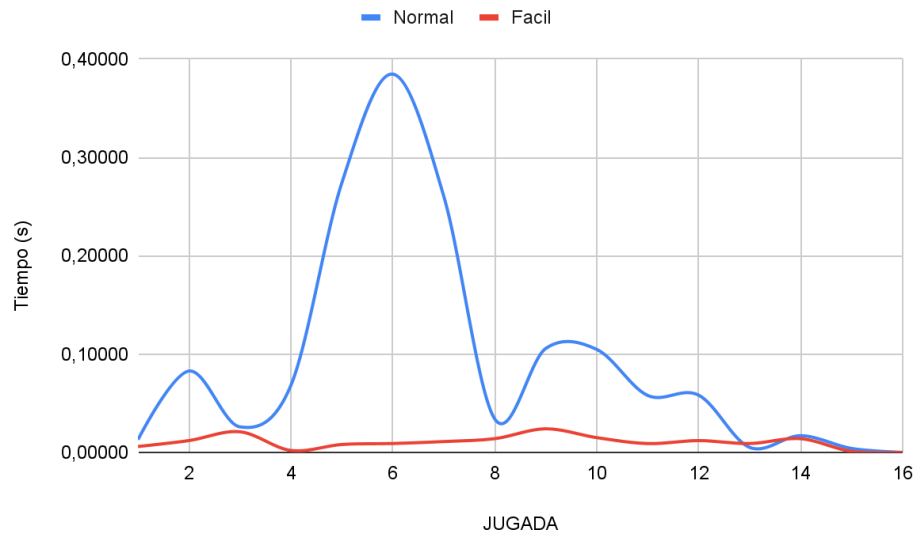


En la apertura o inicio de juego, a pesar de que es el punto de la partida donde existe una mayor cantidad de celdas vacías en el tablero, también es el punto de la partida donde hay menos cantidad de fichas, tan solo las cuatro iniciales, por ende la primera ficha puesta por un jugador (sin contar las cuatro iniciales) tiene una cantidad definida de posibilidades, teniendo sólo cuatro celdas posibles, lo que explicaría que la exploración inicial sea tan reducida.



JUGADA	DIFÍCIL	NORMAL	FÁCIL
	TIEMPO	TIEMPO	TIEMPO
1	0,02597	0,01297	0,00598
2	0,07476	0,08275	0,01201
3	0,81800	0,02596	0,02095
4	0,00896	0,06822	0,00195
5	0,18251	0,27430	0,00802
6	1,36240	0,38476	0,00901
7	2,35864	0,26031	0,01101
8	3,60476	0,03391	0,01396
9	8,00251	0,10572	0,02397
10	0,67139	0,10472	0,01500
11	0,31484	0,05788	0,00897
12	0,04591	0,05817	0,01197
13	0,02198	0,00503	0,00901
14	0,04591	0,01695	0,01392
15	0,01300	0,00398	0,00101
16	0,00097	0,00000	0,00000

En el final del juego, tenemos el caso contrario al anterior, aunque al ir finalizando la partida la cantidad de fichas en tablero son las mayores, esto conlleva a que la cantidad de celdas vacías sean las menores, reduciendo nuevamente la capacidad de la exploración.



En el juego medio o *Mid game*, es donde ambos factores, cantidad de fichas en tablero y cantidad de celdas vacías, tienen una cantidad equilibrada dando una mayor cantidad de posibles jugadas en comparación al inicio de la partida, además se debe considerar que estos factores van variando de tal manera que las posibilidades de tener jugadas posibles aumentan a medida que el perímetro del grupo fichas se acerca a las fronteras del tablero, requiriendo una exploración de nodos cada vez más extensa hasta llegar al cierre de la partida (fin del juego) donde las posibilidades se reducen drásticamente.