Diego García Vázquez

Implementación del algoritmo de búsqueda MINIMAX con poda para dotar a un agente con la capacidad de jugar Othello.

AGENTE JUGADOR DE OTHELLO

Búsqueda entre adversarios con MINIMAX

Índice

[Introducción 2](#_Toc91092714)

[Definición del problema 4](#_Toc91092715)

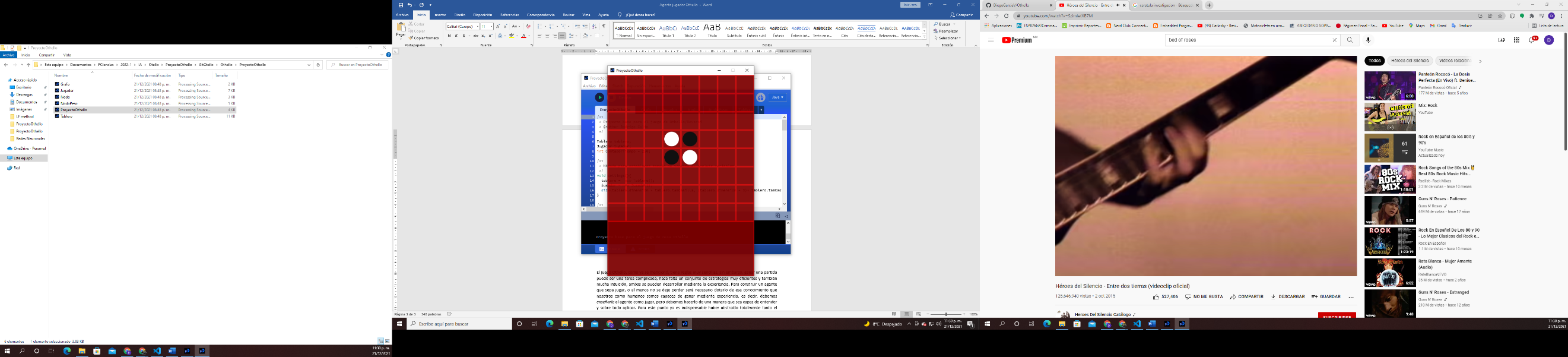
[El entorno 4](#_Toc91092716)

[El problema 6](#_Toc91092717)

[Complejidad de la solución 7](#_Toc91092718)

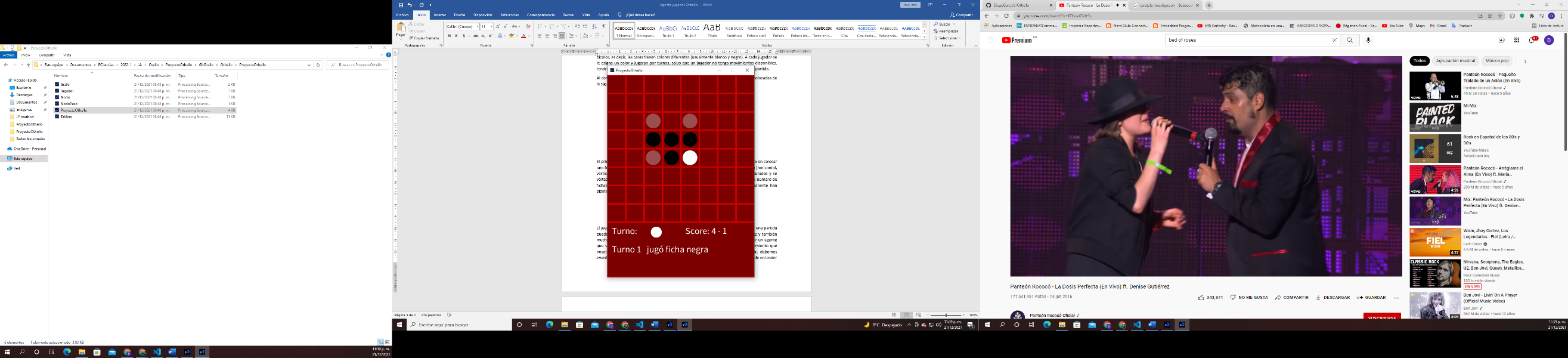
# Introducción

El trabajo aquí presentado tiene el objetivo de exponer el desarrollo de un agente con la capacidad de jugar Othello. Comenzaremos dando a conocer los elementos que conforman al agente, el problema que se le ha planteado, el entorno donde trabajará y lo más importante, como dará solución al problema.

Antes de continuar con la descripción del agente, hablaremos sobre el juego que pretende ganar, el juego Othello es un juego de mesa que de dos jugadores que consta de un tablero de 8x8, 64 fichas bicolor, es decir, las caras tienen colores diferentes (usualmente blanco y negro). A cada jugador se le asigna un color y jugaran por turnos, salvo que un jugador no tenga movimientos disponibles, tendrá que saltar su turno hasta que tenga movimientos disponibles o bien, termine la partida.

Configuración inicial del Tablero

Al comenzar el tablero se encuentra con únicamente 4 fichas, dos para cada jugador, colocadas de la siguiente manera:

El primer turno corresponde al jugador de las fichas negras. Cada movimiento consiste en colocar una ficha en una casilla del tablero que logre encerrar una hilera de cualquier tamaño (horizontal, vertical o diagonal) de fichas del oponente, la hilera o hileras encerradas son capturadas y se voltean, es decir, cambian al color del jugador que realizo la jugada, de esta manera el número de fichas del jugador que ha hecho el movimiento ha aumentado, las fichas del oponente han disminuido y el tablero tiene una casilla menos para realizar un movimiento. La partida termina cuando ningún jugador pueda realizar mas movimientos (Claramente eso incluye el caso cuando el tablero se ha llenado) y gana el jugador que ha posicionado mas fichas sobre el tablero, pudiéndose dar un caso de empate si ambos han capturado 32 fichas. Una observación importante es que el numero de fichas que cada jugador tiene a lo largo de la partida resulta irrelevante a la hora de saber que jugador tiene ventaja o bien, tiene mas posibilidades de ganar, pues en cuestión de muy pocos turnos pueden ser capturadas gran cantidad de fichas.

Jugador 1 tira en casilla [3,4] y captura [4,4]

El juego Othello, como ya se mencionó, tiene reglas muy sencillas, sin embargo, ganar una partida puede ser una tarea complicada, hace falta un conjunto de estrategias muy eficientes y también mucha intuición, ambas se pueden desarrollar mediante la experiencia. Para construir un agente que sepa jugar, o al menos no se deje perder será necesario dotarlo de ese conocimiento que nosotros como humanos somos capaces de ganar mediante experiencia, es decir, debemos *enseñarle* al agente como jugar, pero debemos hacerlo de una manera que sea capaz de entender y sobre todo aplicar. Para este punto ya es indispensable haber abstraído totalmente tanto el problema a solucionar como la manera en que lo hará, una vez que el agente sepa jugar, debemos asegurarnos que sea capaz de interactuar con el entorno, de este modo, nuestro agente podrá cumplir con su objetivo y dar solución al problema, que en este caso es ganar la partida.

# Definición del problema

Para comenzar a construir un agente, es primordial conocer el entorno donde se desarrollará, así como el problema que debe resolver. Dado que buscamos un agente capaz de jugar Othello, no es muy complejo de analizar su entorno, ya que este se limita un tablero de 8x8, sin embargo, el cómo de desarrolla el juego resulta algo menos trivial, pues no basta con conocer las fichas colocadas, más adelante detallaremos una descripción del entorno. Ya que sabemos dónde vivirá nuestro agente, es momento de analizar cuál será su objetivo, es decir, el problema al que dará solución, resulta mucho más sencillo de definir el problema que el entorno, pues el objetivo del agente es ganar, que se traduce en **alcanzar** una configuración del tablero que no tenga más movimientos y se tengan más fichas del agente que del oponente.

## El entorno

Abstraer el entorno para el agente es sencillo en un principio, consta de una matriz de 8x8 (Tablero), donde cada entrada corresponderá al estado actual de la casilla correspondiente, sin embargo, es importante que el tablero admita únicamente un movimiento legal, que conozca el turno del siguiente movimiento, y que se actualice correctamente una vez que se ha hecho una jugada. En pocas palabras, el entorno debe ser consistente con el estado actual de la partida.

Ahora que conocemos que elementos conforman el entorno, podemos comenzar a dar una definición del mismo:

Tablero

**mundo** //Matriz de 8x8 que indica el estado del tablero

**turno** //Valor booleano que indica que jugador debe jugar

**numeroDeTurno** //Valor entero que indica el número de turno actual

**setFicha(x,y)** //Método que realiza movimiento en la casilla

**jugadasPosibles()** //Función que regresa coordenadas de jugadas legales

**esJugable(x,y)**  //Función que indica si la casilla es un movimiento legal

**sinMovimientos()** //Función que indica si hay movimientos para el turno actual

**finPartida()** //Función que indica si la partida ha concluido o no

En el esquema anterior, vemos las características que identifican el estado del entorno y algunas acciones que debe ser capaz de realizar para interactuar con los jugadores. Como podemos ver el tablero indica cuando una jugada es o no posible, de este modo restringe a los jugadores a mover fichas únicamente donde pueden hacerlo.

Ya que hemos abstraído el entorno de trabajo, ahora debemos aclarar como el agente va a interactuar en el, es decir, la REAS. Cuando una persona juega Othello, lo único que necesita es ver el tablero y poder colocar una ficha, de una forma similar, el agente solo necesitará acceder a las propiedades del tablero mediante las funciones y métodos que se definieron con anterioridad, de ahí la importancia de una correcta abstracción del entorno. A continuación se enuncia la REAS de nuestro agente:

* Medida de Rendimiento: Para medir el rendimiento de nuestro agente debemos evaluar la eficiencia con la que desempeña su trabajo, es decir, la eficiencia con la que gana una partida. Los factores que consideraremos para medir la eficiencia del agente son el tiempo de ejecución, el espacio de memoria que utiliza y que tan bien juega (proporción de partidas ganadas).
* Entorno: El entorno ya se ha estudiado y corresponde al tablero de 8x8 y el estado actual del juego (numero de turno y jugador que realizará el siguiente movimiento).
* Actuadores: Dado que se trata de un agente abstracto y que está construido en su totalidad por software, sus actuadores serán funciones o métodos que le permitan interactuar con su entorno. El único actuador que nuestro agente necesita es un método que le permita realizar un movimiento, es decir, colocar una ficha en el tablero en una posición legal, para esto primero tendrá que saber si la jugada es legal, pero eso es tema de los sensores
* Sensores: Al igual que los actuadores, los sensores de nuestro agente serán funciones que consulten el estado actual del entorno, en otras palabras, funciones que tomen como parámetro el tablero y de él obtengan la información necesaria.

## El problema

Ya tenemos claro que problema debe solucionar el agente, alcanzar un estado donde este sea quien gane la partida, es decir, llegar a un estado donde no haya movimientos posibles y el agente tenga más fichas capturadas sobre el tablero, suena sencillo, sin embargo llegar a ese estado no es tarea fácil debido a que no se sabe el movimiento que hará el oponente en el siguiente turno, por ello no bastará con una simple búsqueda entre el espacio de estados, será necesario añadir un factor importante que es suponer que el oponente siempre hará el mejor movimiento posible.

Al suponer que el oponente tomará siempre el mejor movimiento disponible, nuestro agente estará preparado para el peor de los escenarios posibles, y en ese supuesto, el agente también tomará el mejor camino que lo pueda guiar a un estado final que le de la victoria.

Hasta este punto se ha mencionado un factor fundamental para nuestro agente que no hemos descrito con claridad, se trata de lo que consideramos el *mejor movimiento posible.* Al hablar del mejor movimiento posible es necesario definir un criterio que, dados dos estados, nos diga cual es *mejor*. Siendo más formales, es necesario definir un orden en el espacio de estados, lo más lógico sería ordenar los estados según sus posibilidades de alcanzar un estado de éxito, de este modo, el agente dará prioridad a aquellos estados que le ofrecen una mayor probabilidad de éxito, para ello habrá que explorar el camino completo desde el estado actual hasta el estado terminal de éxito.

El orden definido anteriormente puede parecer prometedor, pero como jugadores experimentados de Othello, sabemos que explorar un árbol hasta encontrar un caso de éxito puede ser extremadamente complejo incluso para una computadora, mas adelante estudiaremos la complejidad de expandir el árbol de una partida de Othello, por ahora nos centraremos en cómo evitar expandir hasta el fondo el árbol de una partida. Si expandimos el árbol una cantidad limitada de niveles, no necesariamente encontraremos un estado terminal, de modo que nuestro criterio de orden ya no sirve, eso nos lleva a extender nuestra definición de orden cuando no se alcanzan estados terminales, aquí es donde nos corresponde *enseñarle* al agente como jugar, es momento de aplicar toda la experiencia adquirida como jugadores y dotar al agente de la capacidad de elegir cuál será el mejor camino a seguir, la solución es definir una heurística que le permita al agente seguir un camino que pese a no llevarlo a un estado de éxito, lo acerque tomando en cuenta ciertos elementos del estado, tales elementos serán evaluados por la función heurística, y aquel estado que maximice la función (o minimice en caso de ser un estado donde el oponente debe jugar) será el estado más prometedor para el agente, llevándolo por un camino que le dé más posibilidades de éxito. Es muy importante plasmar de manera correcta dentro de la función heurística la experiencia e intuición que como jugadores nos han hecho ganar partidas, ya que del poder de la función dependerá el poder del agente.

Finalmente, durante la descripción del proceso anteriormente mencionado, se hablo de maximizar (o minimizar) una función de evaluación, por tal motivo, el algoritmo que se empleará para la búsqueda de un camino prometedor será el algoritmo MINIMAX, la idea del algoritmo es la que se ha descrito para el funcionamiento del agente, una búsqueda en profundidad que propaga hacia los nodos superiores los valores obtenidos en los nodos más profundos, tomando máximos o mínimos según la paridad del nivel de búsqueda.

# Planteamiento de la solución

Durante la definición del problema se expuso a que se enfrenta nuestro agente, como se puede abordar el problema y con que, sin embargo poco se ha hablado de nuestro agente, en esta sección detallaremos todo lo referente al agente.

En la sección anterior se menciono la importancia de definir una adecuada función heurística y para establecer un orden en el espacio de estados, mencionamos también la forma en que se recorrería el árbol de estados y se dio un acercamiento al algoritmo MINIMAX, todos estos componentes serán las entrañas de nuestro agente.

## Función de evaluación

Supongamos que tenemos dos posibles movimientos, cada uno conduce a diferentes caminos que al cabo de unos cuantos pasos podríamos tener resultados abismalmente diferentes, la pregunta es, de todos los posibles estados que se pueden alcanzar, ¿Cuál será el más adecuado? y no menos importante, de los dos posibles movimientos que inicialmente teníamos, ¿Cuál de los dos elegir? Actualmente ya sabemos que esa decisión es fácil de tomar si exploramos todas las posibilidades y vemos cual nos conduce a la victoria, pero en lugar de obtener respuestas solo encontramos más preguntas, ¿Qué tan profundo tendremos que ir para encontrar un estado terminal? , Si ambos movimientos llevan a una eventual victoria, ¿Cuál de los dos deberíamos elegir? , Si no tenemos la capacidad de llegar hasta un estado terminal, ¿Cómo sabremos qué camino tomar? Las respuestas a todas estas preguntas se resuelven implementando una función de evaluación. Esta función de evaluación debe indicar que tan optimo es un camino, encuentre o no un estado terminal, es claro que debe tomar en cuenta las estrategias que como humanos llegamos a utilizar, aunque debemos ser cuidadosos, ya que dotar al agente de estrategias que no son del todo buenas, no conseguirá que juegue tan bien como esperamos, por ello es indispensable tener suficiente experiencia en el juego.

La función de valuación propuesta para el agente es la siguiente:

Analicemos parte por parte la función de evaluación. Primeramente la función de evaluación toma como parámetro un estado (o Nodo), recordemos que el objetivo de la función es definir un orden entre los estados del entorno, por lo tanto *evalFunc* está definida sobre el espacio de estados. Veamos ahora la definición de la función, podemos ver que el valor que retorna cambia dependiendo si se trata de un nodo terminal o no, en caso de ser un nodo terminal la función de evaluación es proporcional a la diferencia de fichas entre el agente y el oponente, si el agente gana entonces el resultado será positivo, en caso opuesto será negativo, notemos que si se trata de un empate, el resultado es cero. Un último detalle de la función cuando se trata de un nodo terminal es el factor de proporcionalidad, se ha asignado un valor con la intención de que el mínimo valor de obtener una victoria sea superior a cualquier valor de aplicar la función heurística a cualquier estado, dando prioridad a los caminos que conducen a una victoria dada una profundidad fija.

Cuando se aplica la función de evaluación a un nodo no terminal, esta retorna el valor heurístico de dicho nodo, en este caso, si todos los estados alcanzados hasta una profundidad constante no son terminales, se optara por seguir un camino que parezca más prometedor según nuestra función heurística, nuevamente hacemos énfasis en la importancia de plasmar correctamente las estrategias a seguir en nuestra función heurística.

En el siguiente apartado hablaremos sobre las estrategias a seguir y como plasmarlas en una función heurística.