Universidad Nacional Autónoma De México
Facultad De Ciencias
Inteligencia Artificial, 2018-II

Proyecto 2: *Othello*

Fecha de entrega:

11 de Abril del 2018

Integrantes:

Aguilera Pacho, Franco Eduardo
Martínez Flores, Jorge Yael
Muñiz Patiño, Andrea Fernanda
Sánchez Morales Rodrigo Alejandro

Si no puedes ganar el juego, si no puedes resolver el rompecabezas,
no eres mas que otro perdedor. (Death Note, 2006)

Resumen

Desarrollaremos una inteligencia artificial capaz de jugar *Othello* contra un jugador humano. Se explicará que algoritmo utiliza la IA programada. Los puntos considerados para la planeación y elaboración del proyecto. Haremos un breve analisis de la complejidad de la IA; ya que la complejidad en tiempo y espacio, son los mayores retos que presenta un desarrollador de *Inteligencia Artificial*. De igual forma analizaremos lo elaborado, asi como los inconvenientes y ventajas de usar un agente y un algoritmo para resolver este problema.

1. El juego

"Un minuto para aprenderlo, una vida entera para dominarlo".

En esta ocasión, se propone implementar una IA que juegue y gane, o almenos intente ganar contra un jugador humano en el juego de *Othello*.

El juego consiste en un tablero con 64 casillas, donde inicialmente hay cuatro fichas colocadas al centro, dos blancas y dos negras, formando una diagonal del mismo color. Las reglas son simples:

- Empiezan las negras.
- Un movimiento consiste en colocar una ficha propia sobre el tablero de forma que 'flanquee' una o varias fichas contrarias. Las fichas flanqueadas son volteadas para mostrar el color propio.
- Es obligatorio voltear todas las fichas flanqueadas entre la ficha que se coloca y las que ya estaban colocadas.
- Una vez volteadas las fichas el turno pasa al contrario que procede de la misma forma con sus fichas. Si un jugador no tiene ninguna posibilidad de mover, el turno pasa al contrario.
- La partida termina cuando ninguno de los dos jugadores puede mover. (Normalmente cuando el tablero está lleno o prácticamente lleno).
- Gana el jugador que acaba con más fichas propias sobre el tablero. Es posible el empate.

El reto es que la IA, en una serie de movimientos, logre que haya más fichas del color correspondiente al final de la partida, ganando asi la partida. Esto puede lograrse con una serie de estrategias que debera contemplar el agente que se utilice para lograr la meta.

2. Sobre el agente

Al desarrollar inteligencia artificial, se tienen 4 enfoques principales a seguir, los cuales son:

- Pensar racionalmente.
- Imitar pensamiento humano.
- Actuar racionalmente.
- Imitar razonamiento humano.

La IA propuesta tendrá como base el segundo enfoque. Para efectos del desarrollo del agente, consideraremos actuar racionalmente a tomar la decisión de colocar una ficha para mejorar nuestra puntuación, de forma inmediata o a futuro. Es decir, que colocar la ficha x nos de un beneficio en el juego y que el movimiento tenga sentido.

Modelaremos el problema (el tablero de juego, asi como las fichas), en el cual el jugador humano podrá enfrentarse a nuestra IA, con Un lenguaje de programación y entorno de desarrollo basado en JAVA, diseñado para la enseñanza y elaboración de medios multimedia interactivos, llamado *Processing*. Se podra jugar contra el agente que decidirá qué ficha tirar cuando sea su turno. La propuesta es que se tengan niveles: fácil, medio y difícil.

El agente a usar será un **agente basado en utilidad**. La pregunta natural sería el por que de no usar un agente basado en objetivos. Al tratarse de un juego como lo es *Othello*, no hay un sólo estado final satisfactorio, puesto que sólo basta con tener al menos una ficha más de nuestro color para ganar la partida. Recordemos que tenemos un tablero con 64 casillas. Con lo cual no sería suficiente ni óptimo utilizar un agente de objetivos, pues los posibles escenarios satisfactorios y/o ganadores son demasiados. Por ello, al agente se le agrega una función de utilidad que proyecta el valor del estado y la toma de decisión con base a el valor de dicha función. Estará tomando entonces una decisión racional apegandonos así al enfoque de IA que decidimos utilizar.

Una vez considerado todo esto. Podemos especificar las características del agente basado en utilidad con una tabla REAS (Rendimiento, Entorno, Actuadores, Sensores)

Agente	Rendimiento	Entorno	Actuadores	Sensores
Agente de Utilidad: *Othello*	Ganar la partida.	Tablero de 64 casillas, fichas del jugador y del agenre, además de las recomendaciones sugeridas.	Las fichas blancas del agente.	Las casillas del tablero.

Cuadro 1: REAS.

El actuador: consiste en colocar la ficha en la casilla x y cambiar el color de las fichas encerradas con la ficha jugada, mediante la representación gráfica hecha en processing.

Los sensores: son las lecturas que indican si hay una ficha en la posición x del tablero, si la hay, entonces poder saber de qué color es, en caso contrario, saber si hay posibilidad de colocar una ficha de color c_i donde $0 \leq i \leq 1$ $c_1 = blanco$ y $c_0 = negro$ dependiendo el turno.

2.1 Herramientas

Ahora, para poder lograr el objetivo del agente en el juego, necesitamos hacer uso del algoritmo llamado Minimax. De hecho, la teoría de juegos se basa en dicho algoritmo. Su objetivo es calcular y preveer qué tan beneficioso es un movimiento u otro, según el juego en el que se aplica, Este algoritmo es usado en distintos juegos, desde juegos sencillos, donde las tiradas son fáciles de analizar, hasta algunos donde no es posible calcular todas las opciones de la tirada actual.

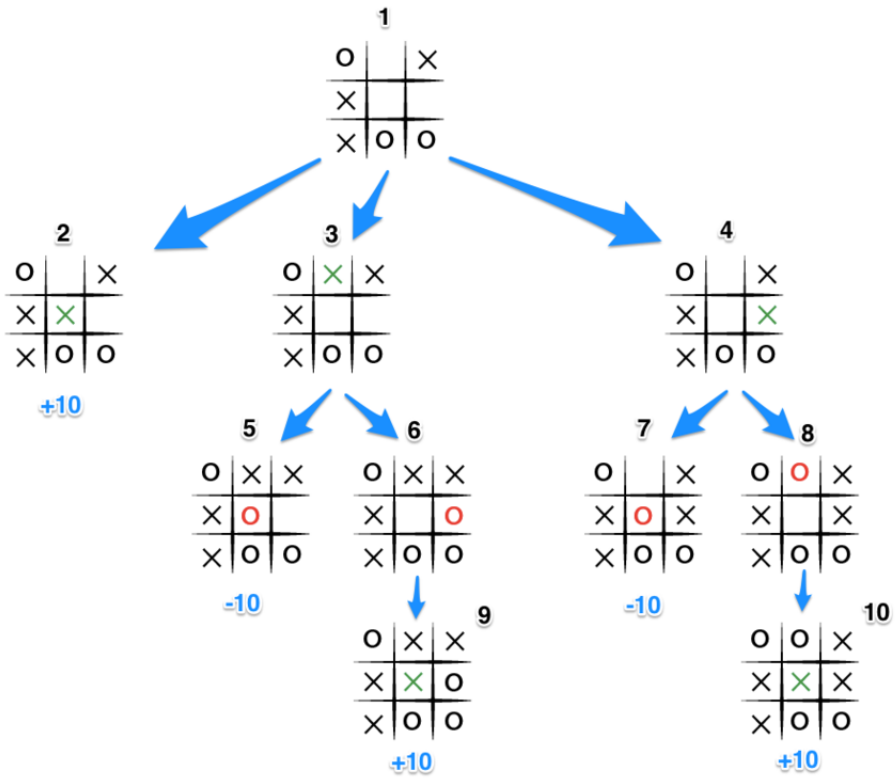

Figura 1: Juegos de *gato* aplicando MiniMax

El algoritmo **Minimax** consiste en lo siguiente: el algoritmo genera de manera parcial un árbol en el que cada nivel representa el turno del jugador A o del jugador B. El árbol siempre considera que el jugador contrario hará la mejor jugada posible, lo que viene siendo el peor caso para la inteligencia artificial. AL considerar esto, la IA tiene que maximizar sus posibilidades de ganar, para que en el siguiente turno minimice las posibilidades del contrincante. Así como se muestra en la siguiente figura:

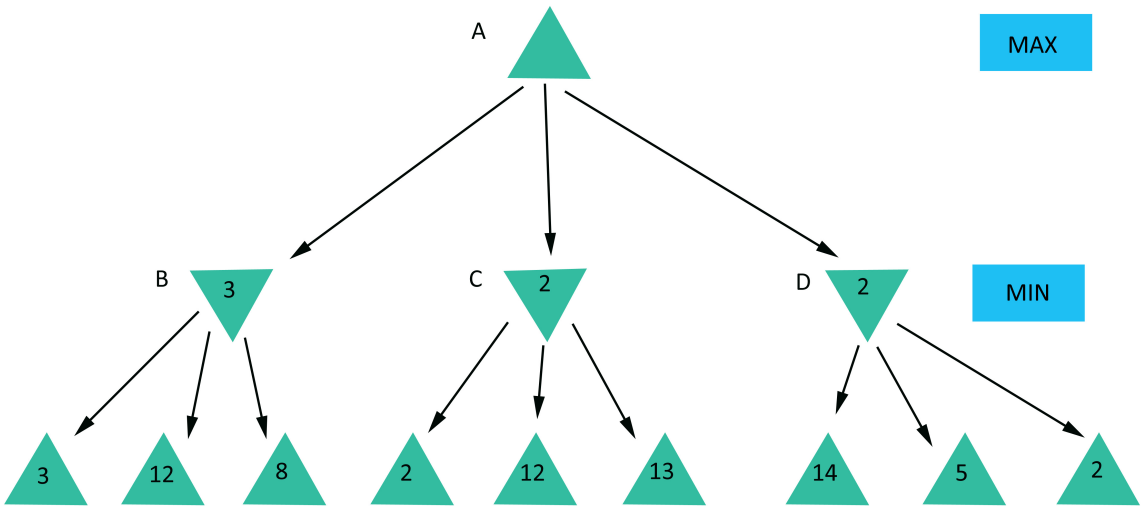

Figura 2: Minimax en el juego del *gato*

Cuando se busca maximizar nuestras posibilidades, se representa con un triángulo como se ve en el primer nivel, al siquiente turno (ya que hemos hecho nuestro movimiento x) queremos minimizar las posibilidades del contrincante.

La desventaja de este algoritmo es que hay que extender el árbol en cada turno, empezando con el nodo raíz como nuestro estado actual, desde el cual se calcula de la mejor tirada siguiente posible. Esto hace que Minimax cree un problema. Al utilizar el recurso de la memoria, puesto que un juego demasiado complejo o con un número muy grande de posibilidades de movimientos y/o puntos a considerar, necesitará una memoria impresionante para poder determinar su jugada, puesto que aún no contamos con alguna herramienta extra.

El hecho de que los juegos que pueden utilizar Minimax para "simular" a un jugador, deben de ser juegos de información completa. Juegos donde se tiene un espacio finito de casillas y n posibles jugadas por hacer. Es decir, está bien establecido el siguiente paso que podemos hacer. aunque esto no sea determinista, sí es posible tener respuestas a incógnitas como:

¿Cúantas casillas quedan libres?. ¿Cúantas fichas negras y blancas hay?
¿En qué casilla puedo colocar la siguiente ficha?
Sí coloco la ficha en la posición x, ¿Dónde podrá poner el oponente la siguiente ficha?.

Es decir, siempre podemos conocer toda la información que involucre al juego y al jugador. Hay juegos donde no es posible hacer esto, por ejemplo, el juego de *piedra papel o tijera*. Ya que el algoritmo no puede contemplar la próxima jugada del oponente, pues esta es totalmente aleatoria y además está el factor de que el "movimiento" es simultaneo, por lo cual el algoritmo Minimax no puede tomar a consideración un "movimiento" anterior.

En la implementacion se ha decidido usar Minimax sin aplicarle ninguna poda con un árbol incompleto. A la par se usa una función de evaluación que es, al menos, la suma de los valores arrojados por 2 heuristicas implementadas. Esta función de evaluación devuelve un valor float grande, cuando el estado favorece al agente (a la inteligencia artificial) y un valor chico si favorece al jugador humano.

3. Entorno de trabajo

Se juega por turnos, sin embargo, en cada uno podemos elegir entre n opciones para encerrar fichas del color contrario. Es por esta razón que no es determinista como tal, aunque en la implementación sí se tiene un orden para establecer qué jugada vamos analizando, esto para tener un orden y no caer en casos repetidos, esto en la creación de ramas del árbol que genera Minimax. Consideremos además que siempre podemos acceder a la información del tablero mediante la matriz, por lo cual en cada momento del juego podemos conocer los datos pertinentes para poder tomar una decisión, haciendo al entorno totalmente observable.

Para determinar nuestro siguiente movimiento lo hacemos mediante la creación de un árbol con el algoritmo Minimax, esto lo hace un agente individual.
En la descripción del juego tenemos que es un tablero de 8x8, por lo tanto nuestro entorno es finito, así como el número de jugadas que hacen ambos jugadores, siendo así, podemos decir que nuestro entorno de trabajo es discreto. La forma del juego hace que nuestro entorno sea episódico, debido a que el siguiente "movimiento" depende totalmente de la acción anterior o dicho de otra forma, del estado actual del tablero.

Entorno de trabajo	Observable	Determinista	Episódico	Discreto	Agente
Juego *Othello*	Totalmente	No	Sí	Sí	Individual

Cuadro 2: Entorno de trabajo

4. Estrategias a considerar

En la implementación se trató de considerar las siguientes estrategias enlistadas, sin embargo no fueron incluidas todas, indicando la prioridad dado que esto complicaba aún mas la función de utilidad de nuestra IA.

- `Esquinas:` Las fichas posicionadas en las cuatro esquinas son las más importantes ya que no pueden ser volteadas.
- `Defender territorios:` tratar de controlar el tablero, al jugar de manera "geométrica¨ con las piezas, creando límites (líneas) donde sólo hay piezas de tu color.
- `Adyacencia:` Los cuadros adyacentes a las esquinas se deben evitar, ya que estas secciones son las entradas para ganar a las esquinas. Con esto queremos decir que se debe evitar colocar una ficha en las casillas adyacentes a la esquina, puesto que el jugar oponente (o inteligencia) puede encerrar las fichas colocando una ficha en la esquina, sin posibilidad de recuperarla.
- `Bordes:` En orden de importancia seguirían los bordes del tablero, debido a que tienen menos posibilidad de ser volteadas. En algunos casos nos pueden ayudar a atacar el interior si se tiene una ficha propia del otro lado del tablero. Pero incluso si no se tiene un control correcto del borde, un jugador más hábil puede hacerse propietario de los bordes del oponente.

- `Guardar movimientos:` Como el nombre lo indica, se deben reservar los movientos que puedes hacer pero que el oponente no puede tocar. Por ejemplo no colocar una ficha en una posición que no hay posibilidad de que el otro jugador use.
- `Cautela al inicio de la partida:` Erróneamente se cree que hacer la movida que siempre nos da el mayor número de fichas es conveniente. Esta estrategia se refiere a que es mejor hacer movimientos que al comienzo de la partida volteen menos fichas contrarias, esto con la finalidad que el oponente tenga menos posibilidades de tiros, de esta manera se tendrán más opciones de jugadas, ya avanzada la partida.
- `Trata de mantener tus fichas en el centro:` Esto con la finalidad de minimizar ataques, de las estrategias anteriores.
- `Trata de mantener tus fichas agrupadas:` Para que de esta forma el rival tenga menos opciones de jugadas.
- `Jugar en los puntos A:` Si vemos como coordenadas el tablero, los puntos A son: (0,3), (3,0) (6,0), (0,6). Ya que si un oponente trata de voltear una de tus fichas en esta posición, mientras controles la tupla correspondiente, lo obligas a darte el acceso a la esquina.

Estas estrategias estan bien para que un jugador humano las tome en consideración, pero computacionalmente se decidio tomar como base algunas de las estrategias y por ello se implementaron 2 heuristicas que son utilizadas en la función.

- **Esquinas**: Cuenta las esquinas capturadas y les da un valor alto, ya que estas no pueden cambiar una vez que una ficha o blanca ha sido colocada.
- **Diferencia**: Número de fichas que difieren entre el jugador y el agente.

4.1 Niveles de dificultad

Los niveles de dificultad de nuestra IA, se implementaron modificando la altura permitida en el árbol que genera Minimax. Esto se especifica con la variable de tipo Integer *profundidad*.

5. Requisitos computacionales

Debido a que usamos un agente basado en utilidad debemos analizar enérgicamente la complejidad de nuestra Inteligencia Artificial. Habrá algunas variaciones en cada nivel de dificultad pero para ello consideraremos siempre el peor caso, para acotar superiormente la complejidad de nuestra implementación.

En este caso tenemos dos complejidades, la teórica y la real, la teórica es la complejidad del algoritmo *Minimax*, y la real es analizar nuestro ejemplar de IA para calcular así el tiempo y espacio que cuesta nuestro agente.

La complejidad teórica del algoritmo minimax corresponde a $O\left(b^m\right)$ en terminos de tiempo y $O\left(bm\right)$ considerando la memoria. Ahora bien la implementación no contempla ningun algoritmo que pode ramas, pero si lo estamos restringiendo en cuanto a la altura del árbol. Lo cual lo hace exponencial, sin embargo al limitar la altura del árbol estaremos en un rango, el cúal podemos permitir tener esta complejidad y al generar menos niveles del árbol, nos toma menos recursos.

Resultados

Ahora nos interesa como es el comportamiento del agente con respecto a su medida de desempeño. El primer punto a mencionar es que el agente no guarda movimiento hechos previamente, de ningún jugador, es decir que no se usa la información para alimentar su analisis. Segundo, el agente todo el momento esta sacando el maximo volador. El agente no tiene precaución de mantener fichas de su color en el centro del tablero, entonces no importa tanto coservarlas, debido a que no le importa conservar territorio si no maximizar su puntaje, sin importarle donde estas las fichas.

5.1 Ventajas y Desventajas

En base a nuestro analisis de complejidad en tiempo y espacio podemos decir que el espacio vendría siendo una desventaja de nuestro agente, pues necesita mucha memoria para tomar la mejor decisión, en cuanto a nuestra medida de rendimiento, esto de acuerdo a su próximo movimiento. Utilizar este agente puede ser poco conveniente en algunos casos más complejos. Otra desventaja de este agente, es que depende completamente de nuestra forma de implementar nuestras heurísticas, siendo que si no cubrimos algún aspecto importante, la heurística dejará de funcionar en cuanto se cruze con ese escenario.

Una ventaja de utilizar este tipo de agente y usar Minimax es que nos permite establecer qué tan "profundo" queremos buscar la mejor opción, de acuerdo con la función de utilidad. Esto nos permite establecer un parámetro para poder proporcionarle un nivel de dificultad al juego, ya que según tan profundo vayamos estamos maximizando nuestras probabilidad de ganar.

Este tipo de agente es conveniente para el objetivo establecido, pues considera los casos posibles, sin error de margen puesto que el escenario es totalmente observable, siendo una ventaja. Y con esto no queremos decir que es el mejor agente para ganar, si no que gracias al algoritmo Minimax considera los posibles movimientos que pueden realizarse y que puede realizar el oponente.

6. Conclusiones

A nuestro parecer no basta con usar algoritmos como Minimax, consideramos que introducciendo otros factores, como por ejemplo la probabilidad y alguna otra función que evalúe, ayudaría a no usar tantos recursos computacionales e incluso esto ayudaría a asegurar que la máquina le gane al oponente humano. Un agente básico no sería capaz de lograr lo que se hizo, pues no tendría la capacidad de considerar los posibles movimientos de oponente, así que usar un agente basado en utilidades proporciona grandiosos beneficios ayudando a contemplar varios parámetros, puesto que la función de utilidad nos ayuda a que el algoritmo de Minimax recorra el mejor camino en el árbol que se genera.

Bibliografía

- Russell, S. & Norman, P.. (2009). Artificial Intelligence: A Modern Approach. Upper Saddle River: Prentice Hall.

- Morillo, A. (2008). Juego de inteligencia Artificial: Othello. Universidad Carlos III de Madrid.
- Wikipedia: Reversi `https://es.wikipedia.org/wiki/Reversi` (Consultado el 7 de Abril del 2018)