



ABALONE
Proyecto Practico De Inteligencia Artificial

Mario German Baquero Cedeño
201667471-3743
mario.baquero@correounivalle.edu.co

Andrés Felipe González Rojas
201759599-3743
andres.gonzalez.rojas@correounivalle.edu.co

Andrés Felipe Medina Tascón
201667602-3743
andres.medina@correounivalle.edu.co

Oscar Alexander Ruiz Palacio
201667600-3743
ruiz.oscar@correounivalle.edu.co

Facultad de Ingeniería
Escuela de Ingeniería de Sistemas y Computación
Programa Académico de Ingeniería de Sistemas
Tuluá, Junio 11 del 2018

Resumen.

Este documento presenta el análisis y diseño del segundo proyecto práctico del curso de Introducción a la Inteligencia Artificial, se implementó un agente inteligente en el juego Abalone usando un algoritmo minimax, se determinó el uso de cuatro heurísticas para el desarrollo de este trabajo y se puede concluir que el uso de múltiples heurísticas determina la capacidad del agente de responder más rápida y eficazmente a situaciones, o jugadas que pueda realizar una persona de tal forma que la maquina sea más versátil ante cualquier situación de juego.

El informe contiene la definición del problema, la manera de como el grupo de trabajo aborda este problema, donde se presenta la implementación del árbol minimax y como se logró hacer. Se analizan las heurísticas implementadas y se realizan las debidas conclusiones del proyecto.



Palabras clave. Min, Max, Heurística, Inteligencia Artificial, Abalone.



A. Introducción

En la construcción de software, especialmente en el diseño de juegos, la utilización de agentes inteligentes es el atractivo de un software interactivo, pudiendo medir las capacidades humanas con los recursos computacionales de la máquina. Los agentes inteligentes tienen la capacidad de percibir entornos, procesar estas percepciones y responder a un entorno sometido de manera racional, maximizando los resultados esperados.

Abalone es un juego de mesa y de estrategia diseñado para 2 jugadores por Michel Lalet y Laurent Levi, en el que cada jugador controla fichas de colores opuestos a las del adversario sobre un tablero hexagonal, las canicas se pueden mover según las reglas del juego, el objetivo del juego es sacar del tablero seis fichas del oponente. Máximo se pueden mover 3 fichas. Esas 3 fichas pueden empujar 2 de su color contrario, de la misma forma 2 fichas solo pueden empujar 1 ficha del color contrario y no se puede empujar fichas en el sentido que estén intercaladas.

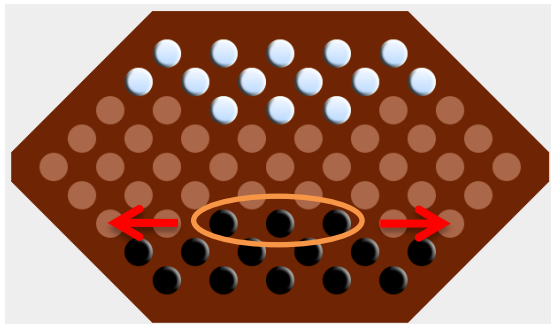
Representación de archivo de texto:

Elemento	Representación	Imagen
Casilla libre en el tablero	0	
Ficha IA	1	
Ficha Jugador	2	
Espacios en el tablero	3	
Espacios a no mostrar	4	
Lados Izquierdo superior	5	
Lados derecho superior	6	

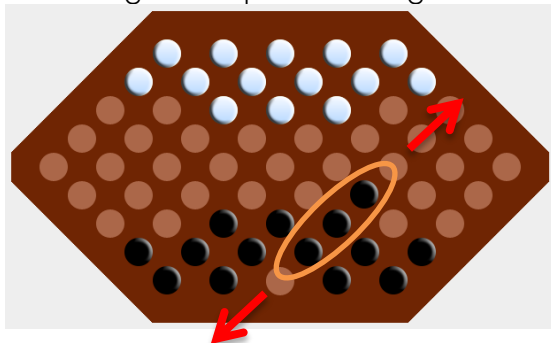
Lados izquierdo inferior	7	
Lados derecho inferior	8	

Durante cada turno, una ficha se puede mover de la siguiente manera (aplica también para las fichas de la maquina):

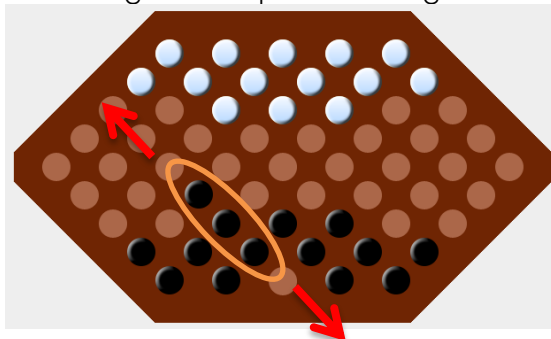
- Hacia los lados



- Línea diagonal 45 positivo o negativo.



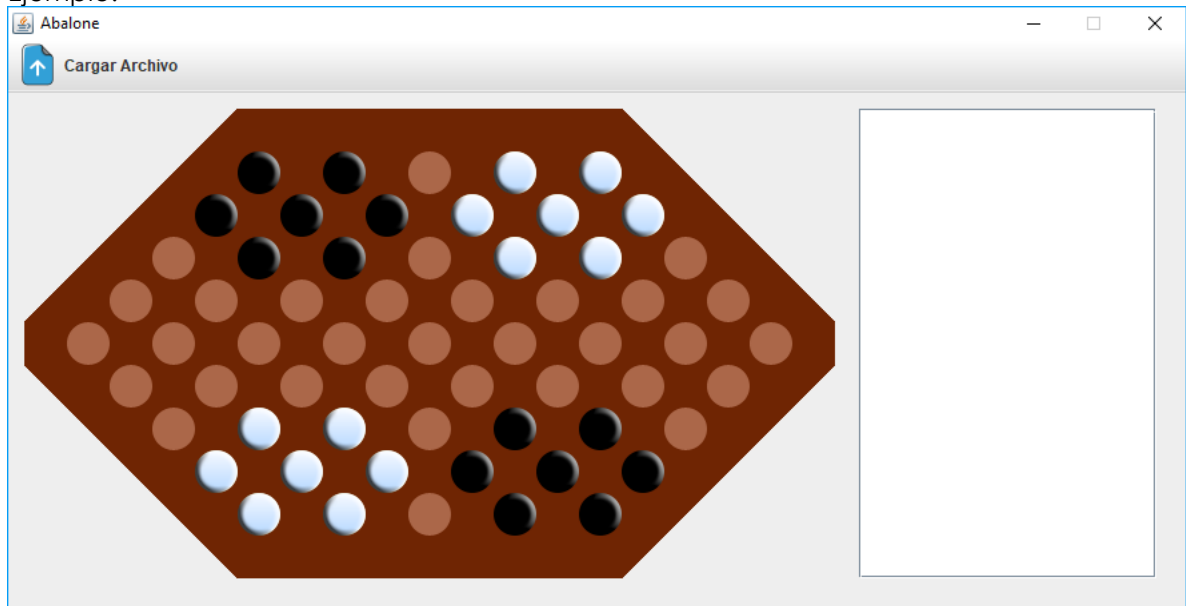
- Línea diagonal 135 positivo a negativo.



El objetivo del juego es sacar del tablero las fichas del adversario, el juego termina cuando alguno de los dos jugadores pierda 6 fichas.

La aplicación debe leer un archivo de entrada con el siguiente formato que representa el tablero.

Ejemplo:



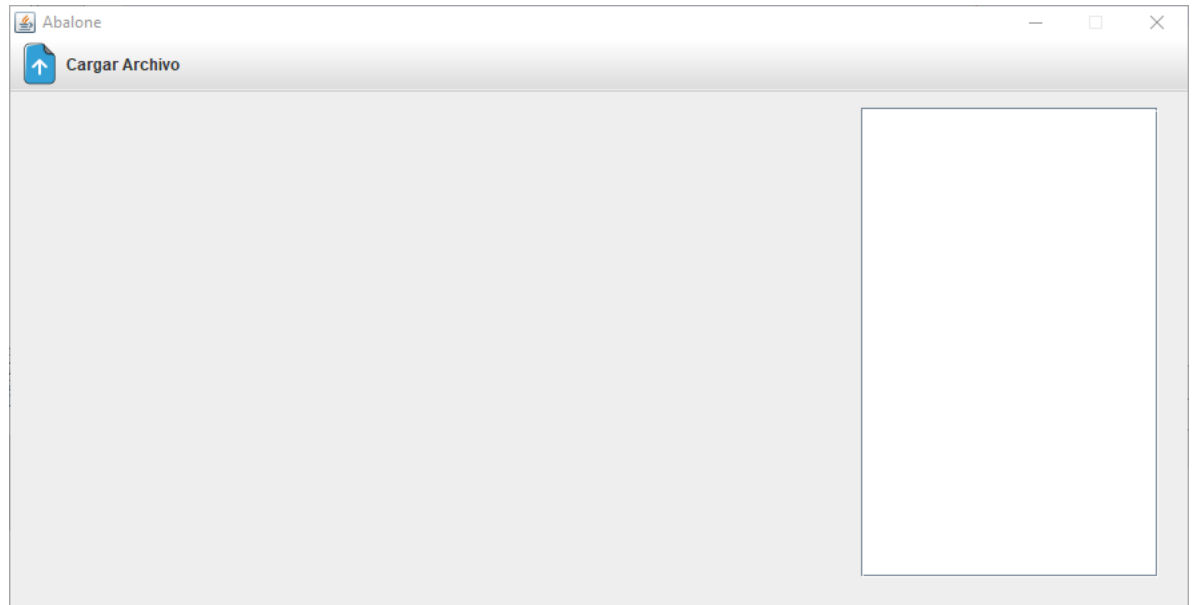
Datos en el archivo:

```
444453333333333364444
4445323230313136444
4453232323131313644
4530323230313130364
5303030303030303036
3030303030303030303
7303030303030303038
4730313130323230384
4473131313232323844
4447313130323238444
44447333333333338444
```

Es recomendable no cambiar los valores en negro, para tener una manera más agradable de ver el tablero.

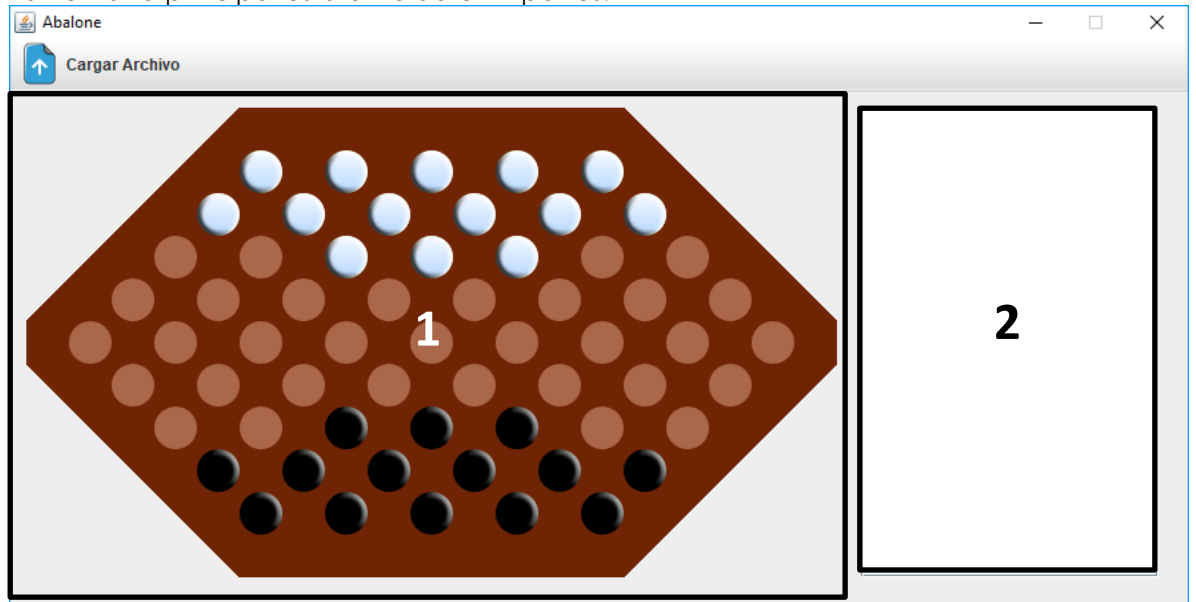
Los valores resaltados con rojo son los únicos posibles valores que se podrían cambiar.

O para mayor comodidad, también se puede cargar un archivo de la siguiente manera:



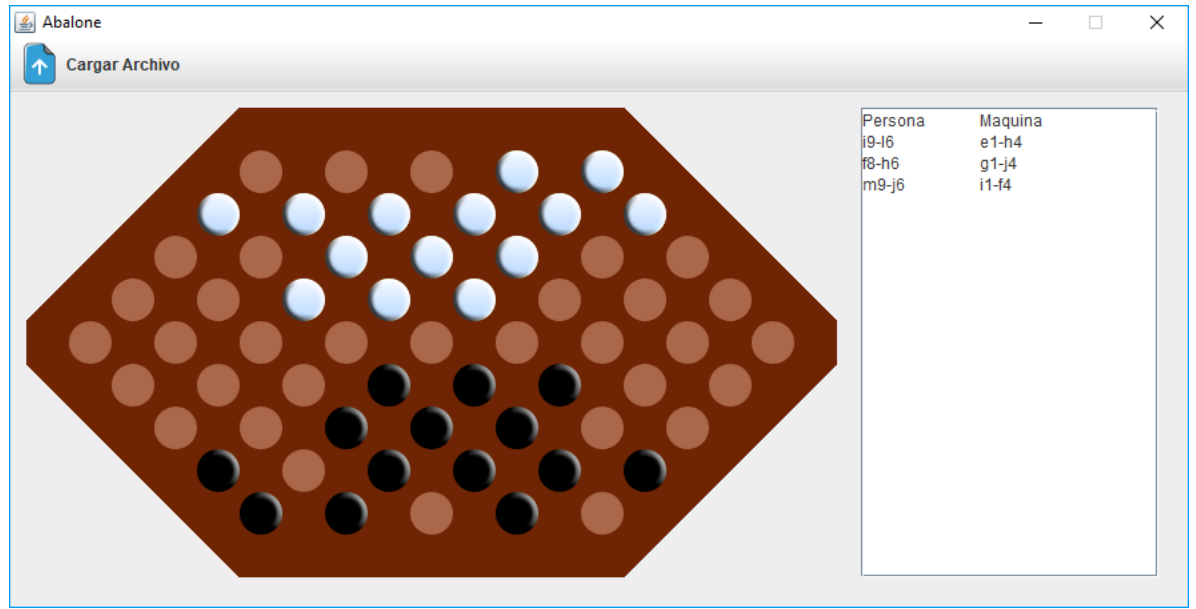
Esta tiene una barra de herramientas en donde se puede elegir el archivo que generara el tablero del juego.

La ventana principal está dividida en 2 partes:



1. Zona del tablero: En esta zona se mostrara el tablero del juego que se genera según el archivo.
2. Zona Información: En esta zona se mostrara las jugadas realizadas, tanto de la persona como de la máquina.

Así se verá la interfaz cuando hayamos elegido un archivo y ya se hayan realizado jugadas de ambas partes.



b. Estructura de la implementación

El proyecto consta en total de 9 clases, las cuales se explican a continuación junto con sus métodos.

Además, cuenta con un paquete llamado Images, el cual contiene las imágenes de las fichas, y diferentes bordes para mostrar de una manera más agradable el tablero.

Abalone: Esta es la clase principal del programa, desde acá arranca la aplicación.

Tablero: Esta clase se encarga de crear la interfaz gráfica del programa y contiene los eventos de todas las fichas, estos eventos corresponden a los movimientos que hace cada ficha para calcular si es posible su movimiento y si es posible empujar rivales.

CargarArchivo: Esta clase se encarga de realizar la carga del archivo de texto plano para que sea visualizado en la interfaz.

Angulo: Se encarga de calcular el valor del Angulo 45 y Angulo 135 para una ficha seleccionada a mover.

Imagen: Se encarga de darle la forma a las imágenes para poder mostrarlas de forma circular.

Posiciones: Es la clase donde se crean las posiciones del tablero con las que se puede interactuar.

Nodo: Es la clase encargada de manejar los atributos de cada nodo, cada nodo contiene un identificador, un tablero, su heurística centro, heurística borde, heurística total, heurística diferencia de fichas, heurística tres en línea, booleano expandido, profundidad en el que se encuentra, jugada realizada y el identificador del padre que extendió.

Movimiento: Es la clase que se encarga de calcular todos los movimientos posibles que puede realizar una ficha a mover y verificar la cantidad de fichas rivales que puede empujar.

Para esto, a partir de su posición, se verifica todas las posiciones referentes al Angulo 45 positivo y negativo, Angulo 135 positivo y negativo y movimiento en el eje X positivo y negativo.

Minimax: En esta clase, se crea el árbol mirando las jugadas posibles de ambas partes con una profundidad de 3, generando la heurística para cada nodo, seguido, se recorre el árbol eligiendo un camino por donde irse con minimax y finalmente, elige una jugada con base a ello.

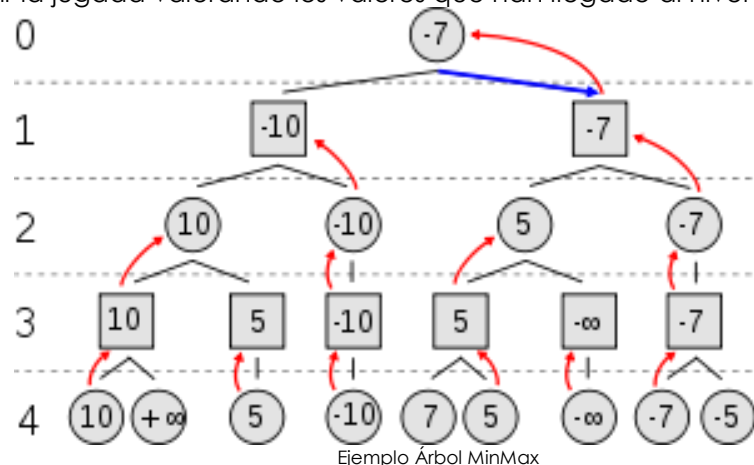
c. Algoritmo Implementado

El algoritmo que se implementó para la realización del proyecto fue el algoritmo Minimax.

El algoritmo Minimax se aplica para el caso de juegos de dos participantes, MAX y MIN. Estos algoritmos permiten encontrar la acción que debería realizar MAX para minimizar la pérdida en el juego.

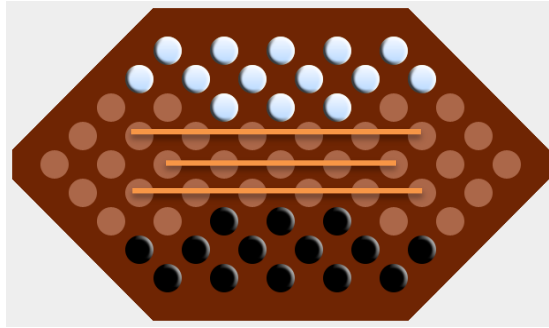
Pasos:

1. Se genera el árbol del juego y se generan los nodos hasta llegar a un estado terminal.
2. Calcula el valor de la utilidad de cada nodo, aplicando 4 tipos de heurísticas implementadas.
3. Calcular el valor de los nodos superiores a partir del valor de los inferiores. Según nivel si es MAX o MIN se elegirán los valores mínimos y máximos representando los movimientos del jugador y del oponente.
4. Elegir la jugada valorando los valores que han llegado al nivel superior.

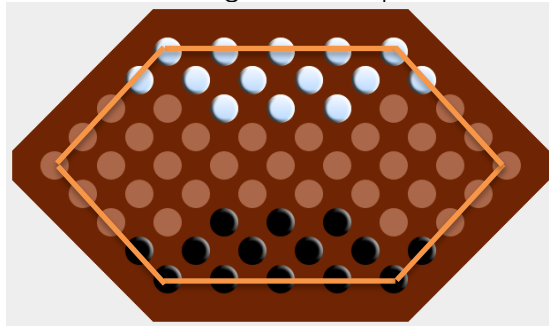


d. Heurísticas

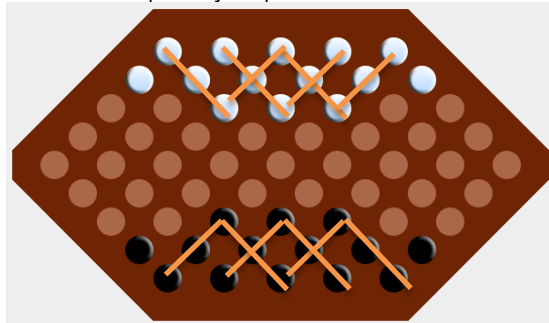
La primera heurística se basa en la diferencia de la cantidad de fichas que se encuentran en el centro del tablero. Esto es debido a que cuando las piezas están más cerca al centro del tablero, menos posibilidades tienen de ser expulsadas fuera, y también a que da más posibilidades de que se encuentren filas de 3, haciendo que sea imposible un empuje



La segunda heurística consiste en la diferencia de la cantidad de fichas que hay en los bordes. Esta heurística se implementó con la finalidad de tener en cuenta cuantas fichas están en riesgo de ser expulsadas del tablero.



La tercera heurística consiste en la diferencia de la cantidad de veces que se encuentran 3 fichas en línea, por ejemplo:



Esta heurística se implementó debido a que es más beneficioso tener fichas en 3 en línea porque da la posibilidad de empujar el máximo número de fichas posibles del rival.

La cuarta heurística consiste en la diferencia de fichas que hay, siendo la cantidad de fichas de la maquina menos la cantidad de fichas que posee el jugador.

C. Conclusiones

El uso de estas 4 heurísticas y darle mayor peso al centro hace que el estilo de juego de la inteligencia artificial realizada sea solido, es decir, intenta mantener la mayor cantidad de fichas en el centro junto con líneas de 3 hace que la maquina intente mantener un centro que no pueda hacer un empuje, lo que conlleva a que no sea

una prioridad para este sacar fichas en del tablero, lo que puede que al enfrentarse con un oponente con un estilo de juego parecido sea un juego un tanto largo a esperar errores, en caso de ser alguien más agresivo puede que castigue rápidamente una agresividad sin sentido o que no tenga reacción a una situación crítica, en caso de pasividad del rival, puede que no la aproveche quedándose estático en el centro manteniéndose allí asfixiando al rival pero sin sacar provecho de ello.

D. Referencias

[1] <https://es.wikipedia.org/wiki/Minimax>

[2] [https://es.wikipedia.org/wiki/Abalone_\(juego_de_mesa\)](https://es.wikipedia.org/wiki/Abalone_(juego_de_mesa))

[3] Diapositivas Curso Inteligencia Artificial, Universidad del Valle Sede Tuluá, Periodo Febrero-Junio 2018.