INFORME CON DESCRIPCIÓN DEL MODELO DEL AGENTE, HEURISTICA, RESULTADOS OBTENIDOS Y CONCLUISONES DEL SEGUNDO PROYECTO DE INTELIGENCIA ARTIFICIAL.

Héctor Fabio Ocampo Arbeláez – Cód. 1355858
hector.ocampo@correounivalle.edu.co
Luis Fernando Quintero Castaño – Cód. 1663714
luis.fernando.quintero@correounivalle.edu.co
Carlos Andrés Riascos Pareja – Cód. 1356541
carlos.a.riascos@correounivalle.edu.co

RESUMEN: *Como requisito del segundo proyecto de la materia Introducción a la Inteligencia Artificial se presenta un documento con lo referente al modelo del agente, la heurística usada para encontrar la mejor alternativa, la poda alfa-beta, expansión de los posibles movimientos y razón para elegir el movimiento de mayor probabilidad de ganar el juego de Linja, también como las conclusiones acerca del desarrollo y resultado del proyecto.*

PALABRAS CLAVE: Modelo del agente, heurística, poda alfa-beta y expansión.

1 INTRODUCCIÓN

Este informe incluye las descripciones acerca del modelo del tablero usado, la heurística y proceso que se llevo a cabo para la construcción del proyecto basado en el juego Linja, el cual se tuvieron en cuenta solamente las siguientes reglas:

-Se hace un primer movimiento (de una casilla) y el segundo movimiento se hace a partir del numero de piezas (propias y contrarias) que había en la franja donde se hizo el primer movimiento (sin incluir la pieza que se mueve).

-Nunca pueden haber más de 6 piezas en una franja (con excepción de las últimas franjas).

-Si con el primer movimiento se alcanza una franja vacía, no habrá segundo movimiento.

2 CONTENIDO

2.1 MODELO DEL TABLERO

El modelo del tablero para el desarrollo del juego de linja consta de objetos utilizados en el código fuente de la siguiente forma:

-Tablero: Es una matriz de tamaño 6x8 donde se guardan las fichas del juego, cada una representada con un número, donde los números del 1 al 12 representan las fichas del jugador Max (en este caso el agente), y los números del 13 al 24 representarían las fichas de Min es decir el jugador humano.

-Posiciones: Es un arreglo de 24 posiciones donde se guardara la posición fila-columna de cada ficha en el tablero de juego.

-FichasFinMax: Es un arreglo donde se guardan las fichas que han logrado finalizar en el tablero correspondientes a Max.

-FichasFinMax: Es un arreglo donde se guardan las fichas que han logrado finalizar en el tablero correspondientes a Min.

-Movimientos: Es un arreglo donde se guarda los movimientos efectuados por la máquina al momento de hacer la expansión.

-ProMovMax: Variable donde guardamos el número de movimientos permitidos para el siguiente turno del jugador Max.

-h: Se trata de la heurística que se calcula al tablero del juego.

2.2 AGENTE

El agente en este caso se divide de dos partes, heurística y expansión.

Heurística: Calcula los puntos que cada jugador tiene en el tablero, los puntos de Max van a ser positivos y los de Min negativos, estos puntos serán sumados y permitirán ver quién tiene la ventaja en el transcurso del juego.

El cálculo se comporta de la siguiente manera: las fichas de Max en la columna 0 tienen un valor de 5, en la columna 1 tiene un valor de 3, en la columna 2 tiene un valor de 2 y en la columna3 tiene un valor de 1. En el caso de Min en la columna 7 tiene valor de -5, en la columna 6 tiene valor de -3, en la columna 5 tiene un valor de -2 y en la columna 4 tiene un valor de -1 respectivamente. Dependiendo del número de fichas que haya en cada columna se multiplica por el valor correspondiente a la columna, las fichas de Max se multiplican en las columnas de 0 a 3 y las de Min en las columnas de 4 a 7.

Ver ejemplo en Anexo 1.

Expansión: Para la expansión se usará el algoritmo mini-max junto con la poda alfa-beta, comienza con un estado inicial sobre el cual Max empezará a hacer los movimientos, la profundidad del árbol es variable dependiendo del número de fichas que estén en el tablero, si encontramos un número de fichas en el

tablero entre 15 y 24 aplicaremos una profundidad 5, fichas mayores que 10 aplicaremos profundidad de 6 y menores que 10 una profundidad de 7.
Razón de la variabilidad de la profundidad: la razón de crear una profundidad variable radica a que cuando se encuentren en el tablero menos fichas la profundidad será mayor lo cual genera que el agente pueda tomar decisiones más precisas cuando el juego esté más cerca del final.

Descripción del algoritmo de expansión junto con la poda alfa-beta:
Aplicamos esta función recursiva donde la condición de parada es la profundidad máxima o un estado final, en el caso que sea una expansión Max o Min, luego se comienza a generar los hijos del estado entrante y comparando su valor heurístico con el de sus propios hijos.
La poda alfa-beta funciona de la siguiente manera:
Ver Ejemplo en Anexos 2.

3 CONCLUSIONES

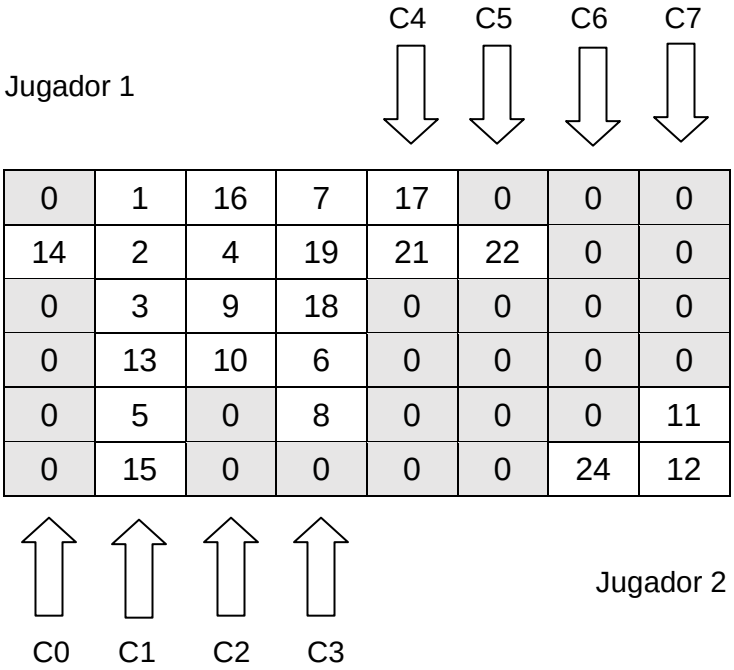
- La inteligencia artificial en el juego linja la podemos ver aplicada en la forma de realizar los movimientos, buscando siempre estar más cerca de la mejor solución poniendo en gran desventaja al contrincante. Además es interesante la forma en que una maquina puede "racionar" antes un determinado estado del juego tomando la mejor decisión para ganar el juego.
- La heurística es muy importante en la inteligencia artificial debido a que por medio de ella es que el agente toma las decisiones respecto a cada estado del juego.
- El algoritmo mini-max es óptimo debido a que por medio de éste se plantea un árbol en el cual podemos ver los movimientos de ambos jugadores, donde la heurística da un valor a la hojas del árbol y en cada profundidad se busca maximizar la posibilidad de ganar el juego o minimizar la posibilidad de ganarlo.
- La poda alfa beta como método de recorte de ramas después de haber aplicado el algoritmo mini-max sirve para ignorar los estados del tablero donde ayudaría a encontrar la mejor decisión, optimizando el manejo de recursos de la máquina.
- El lenguaje de programación python es muy útil al momento de crear algoritmos para inteligencia artificial ya que provee muchas herramientas para el manejo y estructura de datos, además al ser un lenguaje compilado ayuda al manejo de los recursos de la máquina siendo éste más eficiente.

4 BIBLIOGRAFIA

[1] Poda Alfa-Beta, se puede encontrar en:
https://es.wikipedia.org/wiki/Poda_alfa-beta
[2] Poda Alfa-Beta, se puede encontrar en:
http://www.infor.uva.es/~arancha/IA/busqueda/MINIMAX%20y%20poda%20alfabeta.pdf
[3] Algoritmo mini-max, se puede encontrar en:
Diapositivas de clase, campus virtual
[4] Lectura de archivos de texto:
http://pythonya.appspot.com/detalleconcepto?deta=Creaci%C3%B3n,%20carga%20y%20lectura%20de%20archivos%22de%20texto
[5] Uso de listas en Phyton:
http://librosweb.es/libro/algoritmos_python/capitulo_7/listas.html
[6] Uso de librería gráfica para Phyton, se puede encontrar en:
https://wiki.python.org/moin/PyQt4
[7] Algoritmo mini-max, se puede encontrar en:
http://users.dcc.uchile.cl/~jegger/memoria/node36.html

Anexo 1: Ejemplo cálculo de Heurística Agente

Tenemos el siguiente tablero con su respectivo estado:



0	1	16	7	17	0	0	0
14	2	4	19	21	22	0	0
0	3	9	18	0	0	0	0
0	13	10	6	0	0	0	0
0	5	0	8	0	0	0	11
0	15	0	0	0	0	24	12

Para Jugador 2 (Max) que contiene fichas del 1 – 12:
- Como no posee fichas en la C0 y ninguna ha terminado, no multiplicamos ningún valor por 5.
- Como las fichas 1, 2, 3 y 5 se encuentran en la C1, multiplicamos la cantidad de fichas (4) por 3.
- Como las fichas 4, 9 y 10 se encentran en la C2, multiplicamos la cantidad de fichas (3) por 2.
- Como las fichas 8, 7 y 6 se encuentran en la C3, multiplicamos la cantidad de fichas (3) por 1.

Para Jugador 1 (Min) que contiene fichas del 13 – 24:
- Como en el tablero no se encuentran las fichas 20 y 23 quiere decir que han terminado y no se encuentras más fichas en C7, por lo tanto multiplicamos la cantidad de fichas (2) por -5.
- Como las ficha 24 se encuentran en la C6, multiplicamos la cantidad de fichas (1) por -3.
- Como las ficha 22 se encentran en la C5, multiplicamos la cantidad de fichas (1) por -2.
- Como las fichas 17 y 21 se encuentran en la C4, multiplicamos la cantidad de fichas (2) por -1.

Entonces el cálculo queda de la siguiente forma:

Para Max: *4x3 + 3x2 + 3x1 = 21* puntos

Para Min: *2x-5 + 1x-3 + 1x-2 + 2x-1 = -17* puntos

La Heurística calculada sería: *21 – 17 = 4*, debido a que el resultado es positivo, quiere decir que Max lleva la ventaja.

Anexo 2: Ejemplo cálculo de Expansión y poda Alfa-Beta

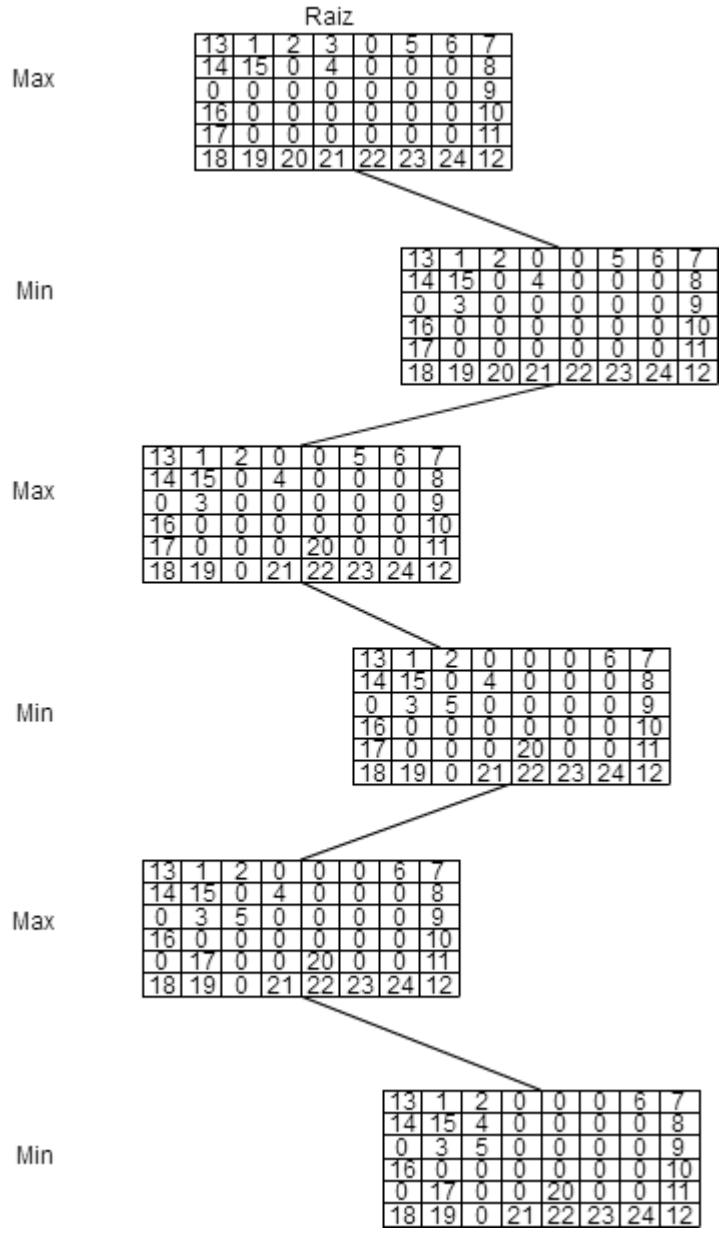
Basado en un seudocódigo encontrando en *https://es.wikipedia.org/wiki/Poda_alfa-beta* definido de la siguiente manera:

```
función alfabeta(nodo, profundidad, α, β, jugadorMaximizador)
    si nodo es un nodo terminal o profundidad = 0
        devolver el valor heurístico del nodo
    si jugadorMaximizador
        para cada hijo de nodo
            α := max(α, alfabeta(hijo, profundidad-1, α, β, FALSE))
            si β≤α
                break (* poda β *)
        devolver α
    si no
        para cada hijo de nodo
            β := min(β, alfabeta(hijo, profundidad-1, α, β, TRUE))
            si β≤α
                break (* poda α *)
        devolver β
```

Donde el llamado a la función lo podemos ver como:

alfaBeta (origen, profundidad, -infinito, +infinito, TRUE)

Ejemplo de Expansión:



Por cada uno de los tableros mostrados en el ejemplo, se desprenden otros 11 tableros indicando distintas jugadas posibles, se calcula la heurística de cada tablero en la última profundidad y de esta manera se elige la jugada.