AJEDREZ EN COMPUTADORA

GENESIS TORRES[1]
EVELIN CHICA[2]
WILLIAM VLADIMIR KOZISCK[3]

UNIVERSIDAD DE GUAYAQUIL
FACULTAD DE CIENCIAS MATEMÁTICAS Y FÍSICAS
CARRERA DE INGENIERÍA EN SISTEMAS COMPUTACIONALES

I. INTRODUCCIÓN

El ajedrez es uno de los juegos de mesa más antiguos y tradicionales que se juegan en el mundo entero, sin distinción de razas, culturas y costumbres sociales. Según el rastreo que se ha hecho acerca del origen de este popular juego de mesa, se ha concluido que es algo así como un pariente lejano tanto del ajedrez chino como del japonés, porque hay una creencia que dice que todos ellos son una inspiración del chaturanga, un juego que se practicaba en la India allá por el siglo VI y que a España llegó gracias a la mediación de los árabes, más precisamente en el siglo XIII hay una primera referencia de el en un libro.

El objetivo del ajedrez es de poner el Rey adverso en una posición que se llama "jaque y mate": el Rey ya no puede moverse en ninguna casilla sin hacerse capturar por una pieza adversa.

Cada vez que el Rey está en jaque (que está amenazado de captura) debe o moverse para escapar a la amenaza, o hacerse proteger por una de sus piezas.

Fig.1 Jaque Mate

- **Antecedentes**

Existen varias causas que motivaron la existencia del ajedrez computarizado, como el entretenimiento propio (pudiendo permitir que los jugadores practiquen y se diviertan cuando no hay ningún oponente disponible), también como herramienta o soporte de análisis, para competiciones entre computadoras de ajedrez, y como investigación o abastecimiento del conocimiento humano.
Sin embargo, y a pesar de la sorpresa de muchos, el ajedrez nos ha enseñado muy poco en lo referente a la construcción de máquinas que proporcionen inteligencia humana, o hacer cualquier otra cosa que no sea jugar prodigiosamente al ajedrez. El funcionamiento de los programas de ajedrez consiste, esencialmente, en explorar un número muy elevado de posibles futuros movimientos y aplicarles una función de evaluación al resultado, mientras que las computadoras de Go desafían a los

programadores a idear nuevos enfoques y estrategias de juego.

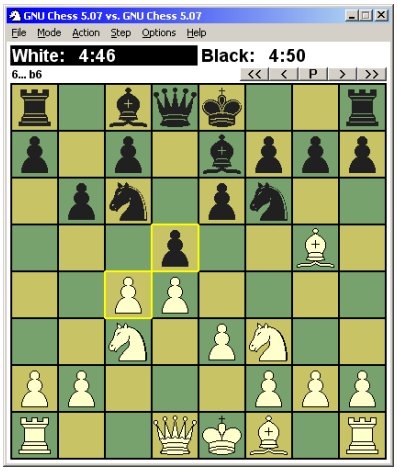


Fig. 2 Muestra de Ajedrez en computadora.

Las tácticas basadas en la fuerza bruta son prácticamente inútiles para la mayoría de problemas que han afrontado los investigadores de la IA. El estilo de juego de un programa de ajedrez se diferencia en gran medida del estilo de juego humano, ya que la elección del movimiento a jugar es totalmente distinta. En algunos juegos de estrategia, las computadoras suelen vencer fácilmente la gran mayoría de partidas, mientras que en otros, los principiantes vencen a las máquinas sin mayor esfuerzo. En el ajedrez, el resultado de la fusión de las habilidades de los expertos, con los programas de ajedrez, es mayor que el de cualquiera de los dos a solas.

II. TRABAJOS RELACIONADOS

- **Minimax**

Imposible generar todo el árbol de búsqueda.

- Generar hasta un determinado nivel de profundidad.
- Aplicar alguna función de evaluación f(N).
  - Devuelve un valor numérico cómo de bueno es un estado.
  - MAX *max*imizará esta función y MIN *min*imizará dicha función.
  - En algunos casos la función nos puede devolver valores como
  - PIERDE, GANA o EMPATA, siempre referidos a MAX.
  - Objetivo del análisis del árbol: determinar valor del nodo raíz (inicio de la jugada). A este valor se le denomina valor MiniMax.

3 en raya

En un tablero 3*3, un jugador posee fichas “X” y otros fichas “O”. En cada turno el jugador coloca una ficha en el tablero.

Gana el que consigue colocar tres de sus fichas en línea.

X	O	
	X	
	X	O

X	O	X
O	X	O
X	X	O

Fig. 6 Muestra de 3 en raya

Implementación

En lo que sigue, asumiremos que en lugar de una función f-utilidad (estado, turno), disponemos de una función f-e-estática (estado, turno), definida sobre todos los estados.

También necesitaremos dos variables: *máximo- valor* y *mínimo-valor* almacenando, respectivamente, cotas para el mayor y el menor valor que puede tomar la función de evaluación estática.

Ejemplo de función de evaluación estática en el 3 en raya

Función de evaluación estatica:

- Una línea potencialmente ganadora para un jugador es aquella que el jugador podría completar para ganar.

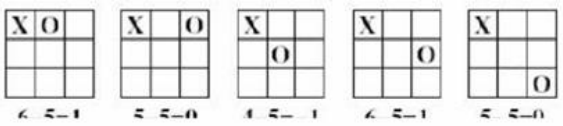
MAXIMO-VALOR=999999 y *MINIMO-VALOR*=-999999

Función F-E-ESTATICA (ESTADO, TURNO)
Si ES-ESTADO-GANADOR (ESTADO, TURNO, MAX), devolver *MAXIMO-VALOR*

Si no, si ES-ESTADO-GANADOR (ESTADO, TURNO, MIN), devolver *MINIMO-VALOR*

Si no, si ES-ESTADO-FINAL (ESTADO), devolver 0

Si no, devolver la diferencia entre el número de posibles líneas ganadoras para MAX y el número de posibles líneas ganadoras para MIN.



Profundidad del árbol de juego

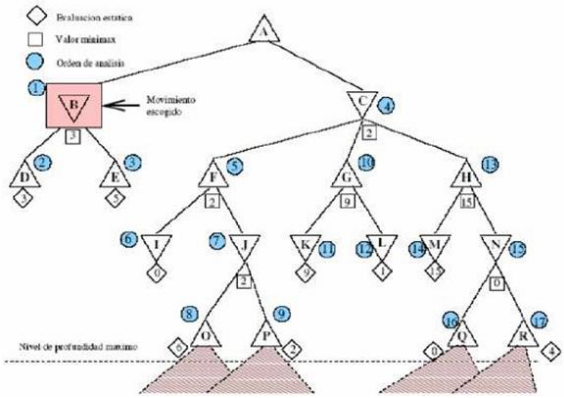


Fig. 7 Árbol de juego.

III. DATOS

- Generación de movimientos

Entre estos tenemos:

Generación Selectiva: (Shannon) Examinar el tablero y obtener una serie de movimientos "buenos" descartando el resto.

Generación Incremental: Generar algunos movimientos, esperando que alguno de ellos será lo suficientemente bueno o malo tal que la búsqueda a lo largo de esa línea de juego pueda ser terminada antes de generar las otras.

Generación Completa: Generar todos los posibles movimientos, esperando que la tabla de transposición contendrá información suficientemente relevante para hacer la búsqueda lo más eficiente posible.

Hardware: DEEP THOUGHT

- MINIMAX
Problema en la mayoría de los juegos, un árbol de juego completo es enorme.

- Complejidad exponencial y tiempo limitado para decidir.
- En la práctica es imposible generar todo el árbol.

Complejidad en el ajedrez

- Número medio de posibles movimientos: 35
- Número medio de movimientos de una partida: 100
- Árbol con $35^{100} \sim 10^{154}$ nodos!!!

Idea:

- No expandir el árbol hasta los estados finales sino solo hasta un determinado nivel de profundidad.
- Valorar los estados que sean hojas del árbol usando una función heurística.
- Propagar valores usando el principio minimax.

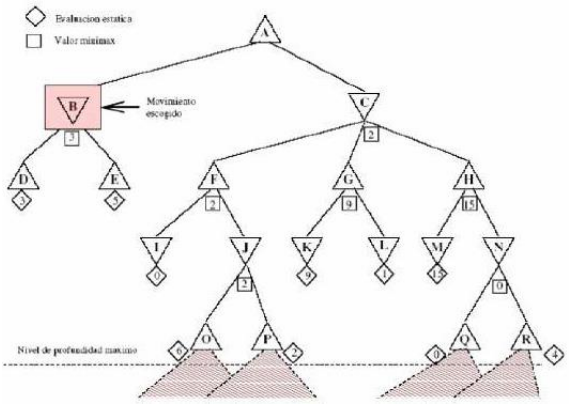


Fig. 8 Árbol de MINIMAX

- Poda ALFA – BETA

Cada nodo se analiza teniendo en cuenta el valor que por el momento tiene y el valor que por el momento tiene su padre.

Esto determina en cada momento un intervalo de posibles valores que podría tomar el nodo.

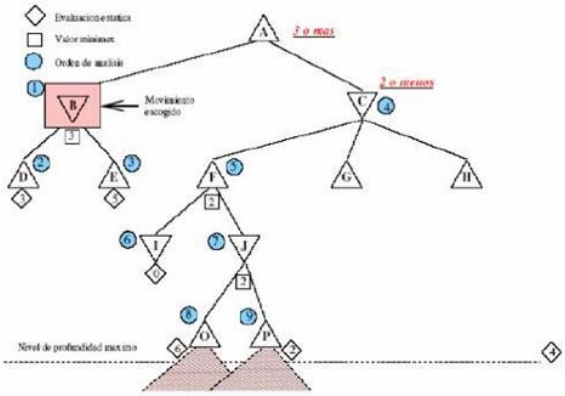


Fig. 9 Árbol de ALFA-BETA

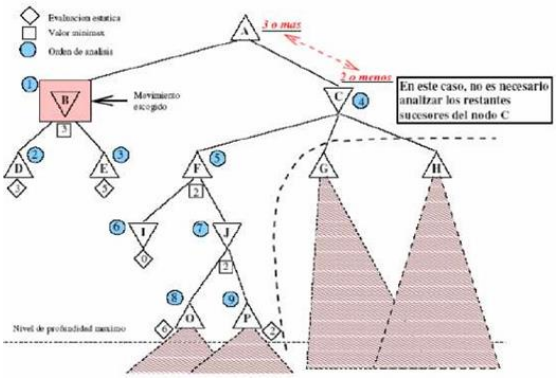


Fig. 10 Árbol de ALFA-BETA

Significado intuitivo de en cada momento:

<u>Nodos MAX</u>: Es el valor actual del nodo (que tendría eso o más) y es el valor actual del padre (que tendría eso o menos).

<u>Nodos MIN</u>: Es el valor actual del nodo (que tendría eso o menos) y es el valor actual del padre (que tendría eso o más).

La poda se produce si en algún momento:

- Y no hace falta analizar los restantes sucesores del nodo.
- En nodos MIN, se denomina poda y en los nodos MAX, poda.

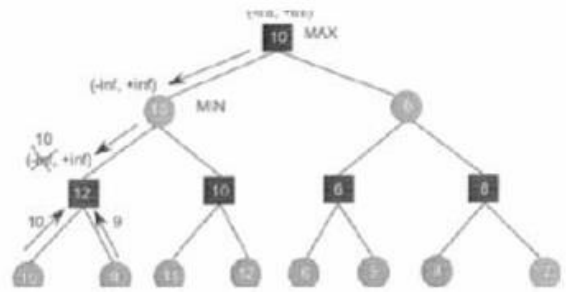


Fig. 11 Movimiento en el árbol 1

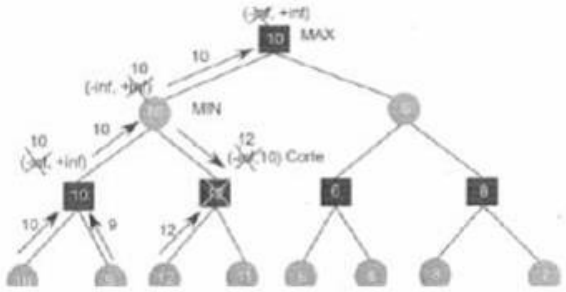


Fig. 12 Movimiento en el árbol 2

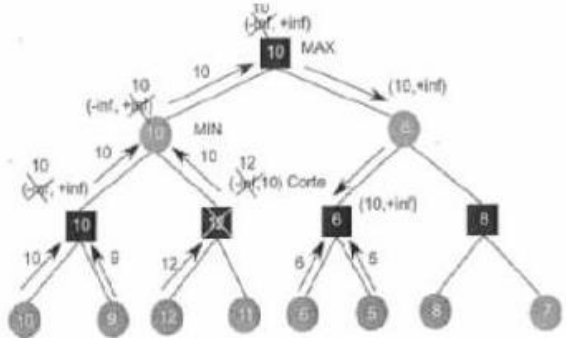


Fig. 13 Movimiento en el árbol 3

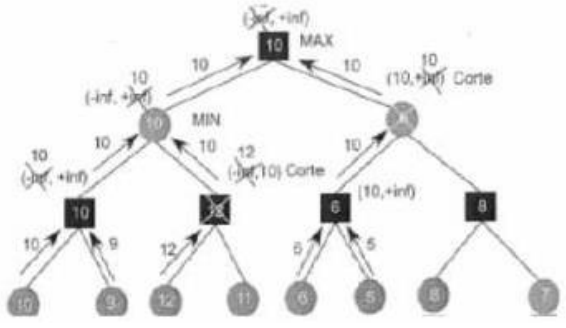


Fig. 14 Movimiento en el árbol 4

IV. METODOLOGÍA

BITBOARDS

Algunos de los bitboard que la mayoría de las computadoras utilizan son:

- 64 mapas los cuales representan las casillas atacadas por cualquier pieza (si hay alguna) que ocupa una casilla en particular.
- 64 mapas que representan, inversamente al anterior, las casillas desde las cuales hay piezas atacando una casilla en particular.
- 2 mapas que representan las casillas atacadas por cada bando (blanco y negro).
- 12 mapas cada uno de los cuales representa las casillas ocupadas por cada tipo de pieza de cada bando (por ejemplo, Caballos blancos).

REPRESENTACIÓN DEL TABLERO: MATRIZ NUMÉRICA

-4	-2	-3	-5	-6	-3	-2	-4
-1	-1	-1	-1	-1	-1	-1	-1
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
1	1	1	1	1	1	1	1
4	2	3	5	6	3	2	4

- Cada casilla toma el valor de la pieza que la ocupa.
- Para mover se cambian los valores de las casillas.
- Falta información: enroque, posibilidad de comer al paso.
- Filas y columnas adicionales: mayor eficiencia en la generación de movimientos.

0 = vacía, 1 = peón, 2 = caballo, 3 = alfil, 4 = torre, 5 = dama, 6 = rey

Fig. 15 Matriz BITBOARDS

La forma básica de cómo generar movimientos con mapas de bits se describe a continuación:

- Utilizar el mapa de bits para todas las piezas por color. Si encontramos que una pieza de ese color está ubicada en la casilla correspondiente sus movimientos deben ser generados. Si no, ir al siguiente bit.

- Determinar si la pieza en cuestión es un peón. Esto se realiza mediante la

operación lógica AND entre el bitmap de la posición actual y el de la ubicación de los peones.

- Determinar las casillas a las cuales la pieza en cuestión puede mover legalmente. Si es un peón el programa inicia la operación con el bitmap de destinos de peones y casillas de captura, sino inicia con el bitmap de casillas atacadas desde la casilla considerada. El bitmap para ubicaciones de piezas de igual color es complementado o invertido con tal de entregar un mapa de las casillas no ocupadas por piezas del mismo color. La intersección de este bitmap con el de los destinos de peones o de ataque entrega el bitmap de casillas a las cuales la pieza puede mover.

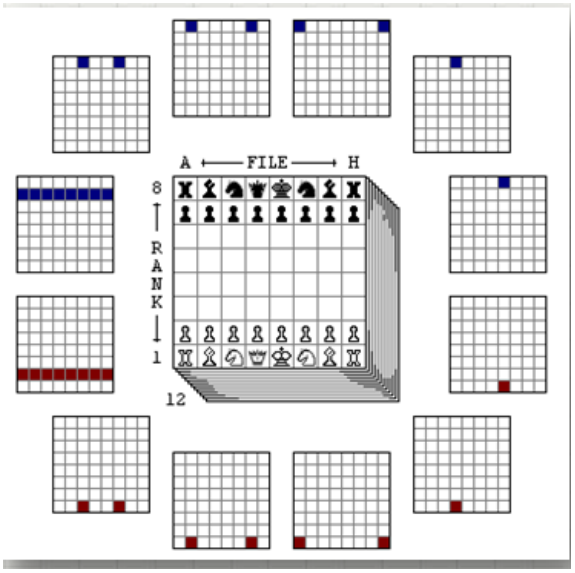


Fig. 16 Posiciones de fichas de ajedrez

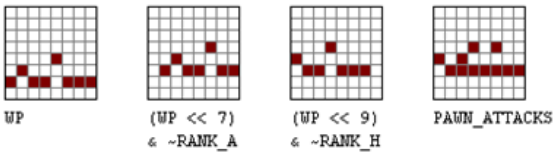


Fig. 17 Posiciones de fichas de ajedrez

Muestra de código para validar movimiento:

```
for (int i=0; i<56; i++)
{       i
        f(board [i] = White_pawn)
        {
                If ((i+1) % 8! = 0)
                pawnAttacks[i+9]= true;

                if ((i+1) % 8 != 1)
                pawnAttacks[i+7]= true;
        }
}
```

V. RESULTADOS

Ordenación

* La eficiencia en la búsqueda bajo Minimax depende del orden de los movimientos en que se realiza esta operación.

* Desafortunadamente, ordenar los movimientos de la mejor forma implica encontrar los mejores y buscar primero sobre estos, lo cual es una tarea bastante difícil de lograr.

Fig. 18 Secuencia de juego

* Por ejemplo, el orden podría iniciarse con capturas, coronaciones de peón (las cuales cambian dramáticamente el balance de material) o jaques (los cuales a menudo permiten pocas respuestas legales), siguiendo con movimientos que causaron recientes cortes en otras variantes a la misma profundidad (denominadas jugadas-asesinas, killer-moves) y entonces observar el resto de los movimientos.

Killer moves

Varias estrategias existen para guardar los movimientos asesinos. Lo más simple es mantener una lista bastante corta delos dos movimientos de profundidad.

Fig. 19 Muestra de juego

En la figura ilustra el funcionamiento de esta heurística. Si es el turno de mover de las blancas éstas intentarían el movimiento 1.Cxh6 debido a que captura la torre. Luego de examinar las réplicas del negro encontrará que este movimiento es refutado por la respuesta 1... Ta1 mate. Entonces, cuando el programa examine nuevos movimientos para el blanco el primer movimiento negro que tomará como respuesta será 1... Ta1 debido a que es un movimiento legal que génerо un corte en una variante anterior.

VI. REFERENCIAS

[1] https://es.wikipedia.org/wiki/Ajedrez_por_computadora

[2] http://www.monografias.com/trabajos83/motivacion-practica-ajedrez-ninos/motivacion-practica-ajedrez-ninos.shtml

[3] http://importancia.biz/importancia-del-ajedrez/

[4] https://en.wikipedia.org/wiki/Deep_Thought_(chess_computer)

[5] https://github.com/Genesis0809/Ajedrez-Gladiador