Algoritmo de búsqueda Minimax con poda Alfa/Beta

Resumen

Se presenta una breve descripción sobre el problema del ajedrez, enfocándonos prontamente a las técnicas para poder diseñar y programar la técnica de búsqueda capaz de efectuar movidas adecuadas ante su adversario. Se especifica la técnica de búsqueda usada para árboles de juego: Minimax, efectuando la poda alfa/beta y la búsqueda Quiescense para optimizar el proceso de búsqueda. Además se especifican las diversas funciones de evaluación que se ha considerado para poder dar una mayor complejidad y mejorar la capacidad analítica de la computadora, ante su adversario, un factor muy importante dentro de las decisiones de juego.

1. Introducción

1.1. El computador y el juego de ajedrez

El ajedrez es un juego que permite retar la habilidad y destreza de los oponentes. Casi gran parte de la capacidad de juego le atribuyen a la rapidez mental de sus jugadores y además la alta capacidad analítica en ese lapso de tiempo para decidir una jugada correcta.

Teniendo en cuenta los factores que definen el éxito en cada jugada como: la rapidez mental y la alta capacidad analítica, entonces, ¿acaso las máquinas podrían jugar el ajedrez?

Esto es una cuestión que ha sido analizada incluso desde que se crearon las primeras máquinas, en la década de 1940. Turing y Shanon comentaron sobre las posibilidades de las máquinas de poder competir con un ser humano. Éste último logró concebir las primeras bases sobre funciones de evaluación para efectuar la mejor jugada, además de parámetros más complejos que iban definiendo un comportamiento de juego, para la máquina, más sofisticado. Entre ellos, Shanon también definió que el comportamiento de las piezas al inicio de juego, no es el mismo que a la mitad de juego y al final. Un claro ejemplo es cuando el rey al inicio de juego no tiene mayor acción, sin embargo, al final de juego, el rey participa activamente matando u obstruyendo las últimas piezas enemigas del juego, además de tratar de no ser capturado.

Un inconveniente sobre este tema, es la relación entre la capacidad analítica del hombre y la capacidad analítica de la máquina. Mientras que el hombre puede deducir heurísticamente las posiciones más importantes del juego, la máquina tiene que efectuar una búsqueda completa de todas las posibles soluciones para poder decidir cuál de ellas, es la más apropiada; esto es lo que se denomina búsqueda de fuerza bruta. No dudamos que la máquina pueda efectuar procesos muy rápidos, sin embargo el ajedrez trae más de un obstáculo para las máquinas. Todo se podría definir con una frase: el ajedrez es infinito.

Nº de Mov.	Cantidad de combinaciones posibles (aprox.)
1	800
2	72 084
3	9,000,000
4	288 000 000,000,000

Tabla 1. Cantidad de posibles juegos en los primeros cuatro movimiento para cada jugador.

1.2 Juego de posibilidades infinitas

El ajedrez mantiene una gran cantidad de jugadas posibles por cada movimiento que se realiza. Sólo mencionar algunas de las características de sus movimientos podremos notar por que se dice que el ajedrez es infinito.
[URL 05] Existen 400 posiciones diferentes al momento que un jugador hace un solo movimiento. Luego de dos movidas para cada jugador existen 72 084 posiciones. Luego 9 millones de opciones al tercer movimiento. Después de la cuarta movida hay 288 billones de combinaciones posibles. [URL 05] Esto lleva a una analogía por parte de Chesmayne: "Hay más composiciones de ajedrez que todas las galaxias juntas".

Luego de estos resultados y analogías, es predecible deducir que es prácticamente imposible, al menos en nuestro tiempo, lograr que una computadora pueda alcanzar todas las posibles soluciones del juego. Por muchos factores como: la capacidad de memoria física, la velocidad y tiempo de procesamiento en relación con el tiempo máximo de espera para poder realizar una jugada en encuentros oficiales de ajedrez.

Por ello se han desarrollado muchas técnicas que han podido se efectivas al momento de permitir seleccionar una jugada buena. Uno de los éxitos más conocidos acerca de ello, es el enfrentamiento de la máquina Big Blue de IBM en 1997 ante el campeón del mundo en dicho año Gari Kasparov. La máquina le venció por 3.5 contra 2.5, obteniendo así la primera victoria de una máquina sobre un campeón del mundo vigente.

En la siguiente sección analizaremos la técnica conocida como Minimax, para la búsqueda de espacio de soluciones. Luego explicaremos las funciones de evaluación utilizadas y posteriormente mostraremos la configuración del juego.

2. Algoritmo de Minimax con poda alfa/beta

2.1 Representación de tablero

La representación del tablero se realiza a través de una matriz numérica.

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

2.1 Estructura Nodo del árbol de poda alfa/beta

La estructura del nodo mantiene la información del tablero actual, del tipo de ficha que está jugando y su posición tanto actual como la posible posición posterior a realizarse, la profundidad que debe explorar las soluciones, la función de evaluación del nodo y el tipo de nodo, ya sea MAX o MIN.

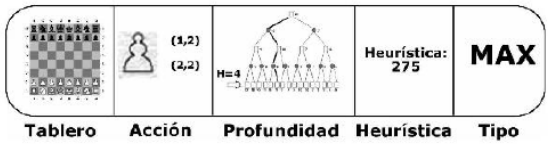


Figura 1. Nodo del árbol con poda alfa/beta

2.2 MVV/LVA

- Los valores se ajustan para que la lista de jugadas quede ordenada de esta forma:
- 1.- Capturas buenas (El atacante vale menos que el atacado, ej. PxD)
  2.- Capturas normales (Valores equivalentes de atacante y atacado, ej PxP) y jugadas que no capturan
  3.- Capturas malas (El atacante vale más que el atacado, ej. TxP)
- Esta forma de ordenar tiene limitaciones evidentes, pero es fácil de implementar, es rápido y permite una poda razonable del árbol de variantes.

El algoritmo de poda se muestra a continuación:

Algoritmo de poda

Entrada:
(NodoAlfaBeta) nodoActual
(Flotante) valor_infinito_maximo
(Entero) maxima_profundidad
(Entero) ident
Salida:
(NodoAlfaBeta) nodoMejorJugada

```
Si nodoActual es MAX
        Funcion_utilidad[nodoActual] ← -VENTANA
Sino
        Funcion_utilidad[nodoActual] ← VENTANA

Si esHoja[nodoActual]
        H ← calcular_heuristica(tablero[nodo])
        Funcion_evaluacion[nodoActual]
        Retornar nodoActual
Sino
        listaFichas ← obtenerListaFichas(tablero)
        Mientras listaFichas no este vacia
                casillaActual ← obtenerPrimero(listaFichas)
                listaNuevosMovimientos←obtenerMovimientos(casillaActual, tablero)
                Mientras listaNuevosMovimientos no esté vacía
                        Jugada ← listaNuevosMovimientos
                ordenarJugada(listaFichas)
                        Actualizar movimiento en nuevo tablero
                        Crear nodoHijo alfa/beta
                        Ident ← ident + 1
                        nodoHijo ← aplicarPoda
                        Actualizamos la información de nodoHijo
                        Actualizamos el valor de utilidad de nodo
                        Cambiar a mejor jugada
Retornar nodoHijo
```

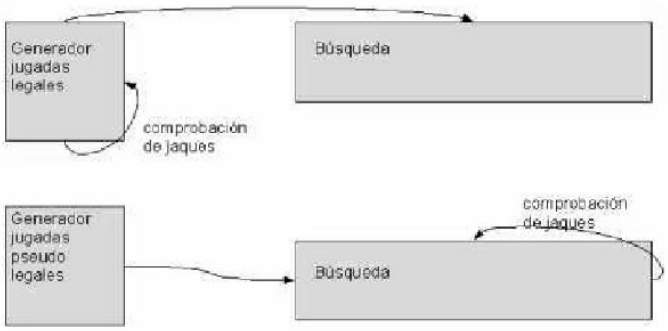


Figura. Se muestra el método MLV para ayudar a la poda alfa/beta

3. Función de evaluación

La función de evaluación es una de las partes más importantes del juego. La función de evaluación permite la posibilidad de medir cuantitativamente las partes positivas y negativas de cierta jugada, es decir permite deducir si la jugada es "buena" o "mala".

Mientras más sofisticada sea la función de evaluación para poder dar una "respuesta inteligente", mayor será también el costo de procesamiento. Esto ha llevado a establecer diversas opciones para funciones de evaluación. En nuestro caso hemos tomado ciertos criterios para poder evaluar nuestras jugadas, las cuales son [URL 02]:

- Evaluación de balance de material
- Evaluación Táctica
- Evaluación de movimientos

3.1. Evaluación de balance de material

En muchos juegos de ajedrez sencillos, ésta, es la única función de evaluación usada, por su simplicidad y aparente efectividad para dar un valor a la jugada de turno. Esta función parte del concepto de que es muy importante y determinante en algunos casos, el poder tener una ventaja numérica respecto al adversario. Si no se logra obtener una ventaja numérica, en el peor caso sería aceptable estar en balance de fuerzas.

Además de ello, no todas las piezas son iguales, es decir, si tenemos solo una ficha en el tablero: como una reina, mientras que el oponente sólo tiene un peón; entonces aún permanecemos en amplia ventaja sobre el adversario, por lo que a cada tipo de ficha se le tiene un valor especial, de acuerdo a su importancia.

Pieza	Puntos	Símbolo
Rey	100	Rm
Dama	9	Dm
Torre	5	Tm

Alfil	4	Am
Caballo	3	Cm
Peón	2	Pm

Tabla 2. Valor de las piezas en el tablero. Parte de la ventaja dentro de un juego se debe al desequilibrio de fuerzas materiales.

El puntaje asignado a cada pieza denota la valía de dicha pieza. A continuación mostramos la fórmula para poder obtener la función de evaluación de material.

Fed = (Nr*Rm + Nd*Dm + Nt*Tm + Na*Am + Nc*Cm + Np*Pm) – (Nre*Rm + Nde*Dm + Nte*Tm + Nae*Am + Nce*Cm + Npe*Pm)

Donde:

Fed.: Función de evaluación de desequilibrio (material)
Nr: número de rey
Nd: número de dama
Nt: número de torres
Na: número de alfiles
Nc: número de caballos
Np: número de peones
Nre: número de rey enemigo
Nde: número de dama enemiga
Nte: número de torres enemigas
Nae: número de alfiles enemigos
Nce: número de caballos enemigos
Npe: número de peones enemigos

Lógicamente una jugada que provoque la pérdida de la reina enemiga generará que la jugada valga más, inversamente si el enemigo inflinge una captura de la reina propia, la jugada tendrá poco valor para se usada.

3.2. Evaluación táctica

Es la función más simple en comparación con las otras dos. Se basa en poder definir ciertos criterios de juego, basado en **penalizaciones** o **bonus**, para los que infrinjan o alcancen estos criterios.

Evento	**Bonus(+)/ Penalización(-)**
Peón obstruyendo a pieza amiga	-20
Peón obstruyendo a enemigo	+20
Peón retrasado con respecto a sus compañeros	-2

Tabla 3. Valores de las penalización y bonus.

La fórmula es:

Fet = nPAObs*-20 + nPEObs*20 - nPR*2

Donde:
nPAObs: Número de piezas amigas obstruidas
nPEObs: Número de piezas enemigas obstruidas
nPR: Número de peones retrasados con respecto a sus compañeros

3.3. Evaluación de movimiento

Es tal vez una de las funciones más importantes, permite definir las consecuencias de realizar un movimiento. Algunas veces realizar un movimiento X podría parecer buena opción, sin embargo muchas veces puedes resultar que dos movidas posteriores, haya sido una decisión catastrófica que pudo haber producido la derrota o considerable pérdida de material.

Esta función a diferencias de las demás, es una función de carácter local con respecto a un nodo. No se evalúan superficialmente, sino que se hace una búsqueda de posibles consecuencias, para cada nodo, al realizar algún movimiento.

Además de ello, la evaluación de movimiento califica a cada pieza, de acuerdo a su tipo, por ejemplo, no es común que una reina, rey o torre, comienzan a jugar a partir de la movida inicial, se sabe muy bien que las movidas iniciales están dadas usualmente por los peones, mientras que las demás piezas se guardan para la mitad de juego o el final.

También mide la capacidad de libertad de movimiento para cada pieza, capacidad de moverse sin ningún ataque de por medio.

Pieza	Puntos	Símbolo
Rey	1	Rmv
Dama	2	Dmv
Torre	3	Tmv
Alfil	4	Amv
Caballo	5	Cmv
Peón	10	Pmv

Tabla 4. Valor de movimiento para cada pieza. Nótese que los peones tienen mayor preferencia de movimiento.

$$\textbf{Fem} = \sum_{i=0}^{np} \text{ncl} + \sum_{i=0}^{np} \sum_{j=0}^{nc} TPmv_{ij} * v$$

Donde:

Np: Número de piezas propias
Nc: Número de casilleros alcanzables para atacar a piezas enemigas
Ncl: Número de casilleros libres para la pieza i.
TPmvij: Tipo de pieza de movimiento, (refiriéndose a los valores de la tabla 2), de la pieza i en el casillero j.
V: valor en función de ataque o defensa.

Algoritmo para calcular V:

Algoritmo Calcular V

```
V ← mv_t
Para cada pieza Pi del tablero actual hacer
    Para cada posible casillero Cj donde puede moverse legalmente
        Si Pi ataca en casillero Cj
            V ← V + mt_te
        Sino
            ncl ← ncl + 1
        Si casillero Cj recibe ataque de represalia
            V ← V – (mt_t)
Retornar V
```

Donde:

mt_te: Tipo de pieza (tabla 2) enemiga capturada en el ataque
mt_t: Tipo de pieza (tabla 2) que después de atacar es capturada

3.4. Función de evaluación

Actualmente la configuración de nuestra función de evaluación es:

$$F = fed + fem + fet$$

4. Efecto Horizonte.-

El efecto horizonte lo podemos describir como la incertidumbre de saber, con que nos podemos encontrar más allá del "horizonte"(ultimo nivel del árbol de poda alfa beta), con lo que nos podemos encontrar nos referimos al hecho de que por ejemplo en el nivel horizonte una jugada del tipo captura sea buena, para un jugador pero solo en ese momento , porque no se ve lo que puede venir después. En la figura 2 se puede observar un ejemplo.

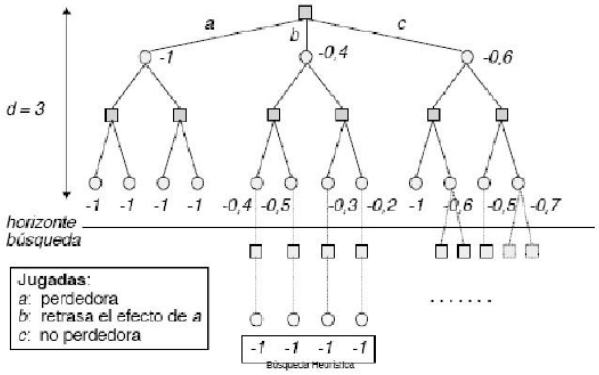
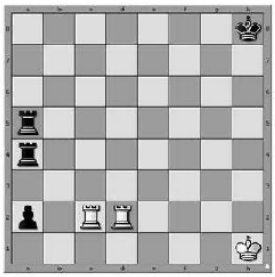


Figura 2. Árbol de poda alfa/beta

Considerando [-1,1]
-1 pierde cuadradito
1 pierde redondo

Si nos damos cuenta la jugada 1 no se escogería por que se iría al pierde entonces supuestamente la siguiente mejor jugada seria la b, pero si antes la analizamos vemos que a una cierta profundidad, se vuelve jugada perdedora, luego la jugada c seria la que se debería realizar, pero si no implementamos un método que pueda manejar este caso entonces, nuestro ajedrez podría no ser muy "inteligente"

Otro ejemplo, viendo ya lo que tiene que ver con jugadas en verdad veremos a continuación:



Aquí podemos ver que una jugada buena según una simple búsqueda seria, torre blanca por peon, pero si vamos más allá de eso vemos que para dos jugadores, normales ocurriría lo siguiente:

torre blanca por peon negro
torre negra por torre blanca
torre blanca por torre negra
torre negra por torre blanca

Es decir por querer ganar un peón nos sacaron ventaja de una torre, esto es el efecto horizonte. Antes de pasar a ver la búsqueda quiescense necesitamos saber algo previo

Posición Estable.- Una posición estable la vamos a definir como que ya no existen mas jugadas del tipo captura en el tablero o mediante un corte luego de recorrer cierta profundidad, en la búsqueda quiescense.

Search Quiescense.- Este nuevo método de búsqueda como ya hemos dicho anteriormente se encargara de hacer una búsqueda de jugadas de solo el tipo captura, de los nodos del nivel horizonte del árbol de poda alfa beta. He ira desarrollándose hasta que sea el nodo desarrollado sea una posición estable.

A continuación proponemos un algoritmo para este tipo de búsqueda

---

Algoritmo Busqueda Quiescense

```
Requiere: jugada //Arreglo de jugadas del ultimo nivel
Devuelve: numero entero, //que indicara que nodo será tomado en cuenta para la sig jugada
        JugadasHorizonte←GetJugadasCaptura()
        Si JugadasHorizonte==vacio
            Return setUtilidad(jugada)
        Sino
                mientras <> vacia
                    jugada=obtenerjugada(i)
                    Si Evaluacion(jugada) == jugada captura
                                return Alfa-Beta-Modificado(jugada)
                    Fin_Si
                Fin_Mientras
        Fin sino
```

Algoritmo Alfa-Beta-Modificado

```
Requiere: jugada
Si situación considerada como empate
    Devolver 0
Si situación es ganadora,
    Devolver mayor-numero
Si situación es perdedora,
    Devolver menor-numero
Si Profundidad = Profundidad-estable
   Devolver evaluación (situación)
Si nivel-max-profundidad
   Para los sucesores cuya situación sea una jugada tipo captura y Alfa < Beta
        Alfa-beta ← ALFA-BETA-Modificado(sucesor,Alfa,Beta,Profundidad+1)
        Alfa ← max(Alfa,Alfa-beta)
        Devolver Alfa
Sino
       Alfa-beta ← ALFA-BETA-Modificado(sucesor,Alfa,Beta,Profundidad+1)
       Beta ← min(Beta,Alfa-beta)
Fin_Si
Devolver Beta
```

5. Configuración del juego

Luego de haber explicado los algoritmos principales que tienen que ver con "la forma de razonamiento de la máquina", mostramos en la siguiente tabla el nivel de dificultad del juego, con respecto a la profanidad del árbol.

Profundidad	Nivel
4	Básico
6	Intermedio
8	Avanzado

Tabla 5. Niveles de dificultad

5.1 Pantalla de juego

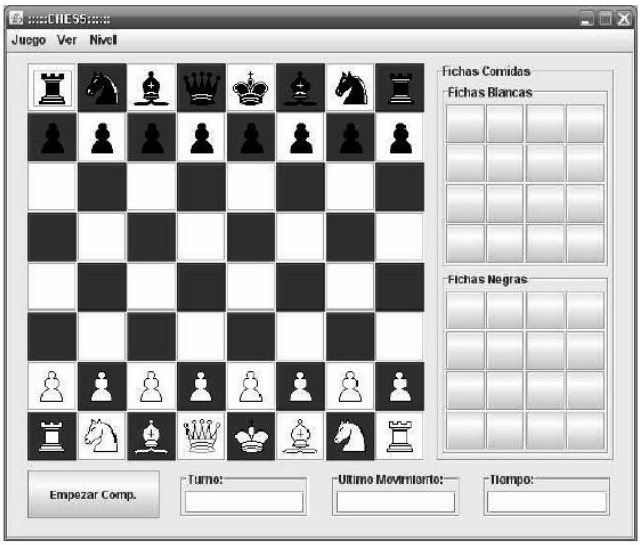


Figura 4. Inicio de juego