Árboles de Juegos

Nelson R. Becerra Correa

Resumen

El presente artículo hace una descripción sobre los árboles de juegos, se presentan los conceptos básicos y se reportan los resultados de un trabajo de investigación sobre el procedimiento *minimax* con corte alfa-beta aplicado a dichos árboles.

Introducción

La inteligencia artificial es una de las ramas de la ciencia de la computación de mayor auge en la última década. El principal objetivo de la Inteligencia Artificial no es reproducir la inteligencia humana sino la investigación de procesos simbólicos, el razonamiento no algorítmico y la representación del conocimiento.

El tema tratado en este artículo tiene que ver con dos aspectos que han dado lugar a la mayor investigación sobre Inteligencia Artificial en las dos últimas décadas; los **juegos** y los **algoritmos de búsqueda**.

Para su desarrollo se va a caracterizar un proceso de juego entre dos adversarios que discurre sobre un tablero reglamentario con piezas que se denominaran **blancas** para el jugador que comienza el juego y **negras** para su adversario.

Los dos jugadores alternan sus movimientos sobre el tablero, de modo que en cada oportunidad sólo se mueve una pieza (salvo excepciones como el enroque en el ajedrez), estando los movimientos regulados por las leyes que excluyen el azar. El juego finaliza cuando se alcanza alguna situación prevista en las leyes.

Las características mencionadas se aplican a juegos como ajedrez, damas, go y mancala entre otros.

El planteamiento que se hace es conducente a posibilitar la construcción de programas que tomen el lugar de cualquiera de los adversarios.

Para el desarrollo de estos programas, se utilizará el procedimiento *minimax* y una mejora de éste conocida como corte alfa-beta, el cual permite generar un árbol de búsqueda para representar el juego y elegir la mejor jugada en un momento determinado, sin tener que examinar todas las posibles jugadas.

El artículo está organizado de la siguiente manera: en la segunda sección se trata la representación de un juego mediante un árbol de juegos; en la tercera sección se trabaja el procedimiento *minimax;* en la cuarta se desarrolla el procedimiento *minimax* con corte alfa-beta; en la quinta se plantea la construcción la realización de un programa de juegos; y en la sexta se presenta el desarrollo de una investigación sobre el procedimiento alfa-beta.

I. Representación Arborescente

La manera natural de representar lo que puede suceder en un juego es mediante lo que se conoce como **árbol de juego**[1], que es un tipo especial de árbol semántico en el que los "nodos" representan configuraciones de tablero y las "ramas" indican cómo una configuración puede transformarse en otra mediante un sólo movimiento.

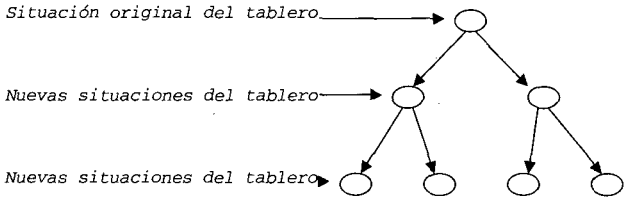

Figura 1. Arbol de Juego

En la Figura 1 se hace una representación en forma de árbol de juegos de uno cualquiera (determinístico, de dos oponentes, con información perfecta y de suma cero) hasta un segundo nivel de profundidad. A continuación se definen unos conceptos básicos sobre arboles[2] .

[1] El concepto árbol de juego se toma de Rusell, ver [3].
[2] Los conceptos de arboles están tomados de Cairó, ver [2]

Ingeniero de Sistemas , MIS Universidad Nacional de Colombia
Profesor Tiempo Completo, Universidad Distrital Francisco José de Caldas

- Un árbol es una estructura jerárquica aplicada sobre una colección de elementos u objetos llamados nodos; uno de los cuales es llamado nodo raíz.

- Un nodo X es descendiente directo de un nodo Y, si el nodo X es apuntado por el nodo Y. En este caso es común utilizar la expresión X es hijo de Y.

- Un nodo X es antecesor directo de un nodo Y, si el nodo X apunta al nodo Y. En este caso es común utilizar la expresión X es padre de Y.

- Se dice que todos los nodos que son descendientes directos (hijos) de un mismo padre, son hermanos.

- Todo nodo que no tiene ramificaciones (hijos), se conoce con el nombre de terminal u hoja.

- Grado es el número de descendientes directos de un determinado nodo. Grado del árbol es el máximo grado de todos los nodos del árbol.

- Nivel es el número de arcos que deben ser recorridos para llegar a un determinado nodo. Por definición la raíz tiene nivel 1.

- Altura del árbol es el máximo número de niveles de todos los nodos del árbol.

II. Procedimiento minimax.

Para ilustrar los árboles de juegos y a manera de ejemplo vamos a seguir la convención de que jugaran las blancas cuando el tablero está representado por un círculo figura 2 (siendo tal situación originada por una jugada inmediatamente anterior de las negras) y que las negras deberán jugar a continuación (situación planteada por una jugada inmediatamente anterior de las blancas).

En un momento cualquiera le corresponde jugar a las blancas. La situación es representada por la figura 2

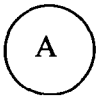

Figura 2.

Lo natural en este momento es que las blancas identifiquen todos los movimientos legales que pueden ejecutar. Esto dará, por ejemplo, el siguiente árbol - ver figura 3 (solo tres jugadas legales permitidas para el blanco)

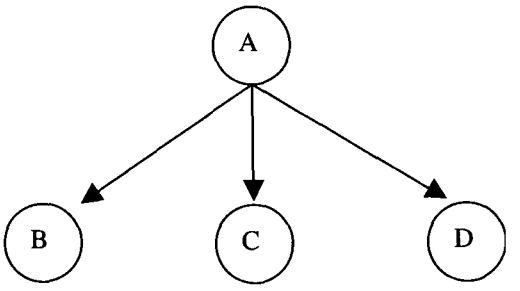

Figura 3.

Siguiendo el esquema natural de análisis, ahora el blanco va a preveer lo que el jugaría nuevamente en cada caso, Figura 4, (asumiendo la perspectiva de las negras).

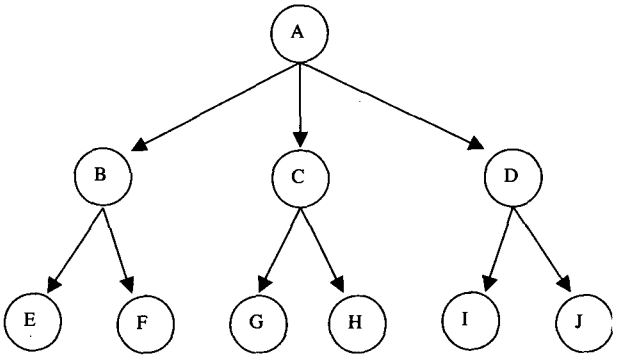

Figura 4.

La exploración de las blancas, figura 4, da como resultado los nodos E,F,G,H,I,J, correspondientes a un tercer nivel de profundidad, (estos nodos plantean una jugada en el tablero A). La exploración podría continuar tanto como la concentración del jugador se lo permita, pero vamos a suponer que se detiene aquí.

En este punto, el blanco, quiere saber cual de los tableros E, F, G, H, I o J es el que conviene alcanzar.

Inmediatamente se recurre al uso de algo que se conoce como la **función de evaluación estática**[3] la cual deberá proveer un valor único para cada tablero, que sea indicativo de su valor para uno u otro jugador.

Una primera forma de imaginar tal función es recurrir a un juego concreto, por ejemplo el ajedrez:

Pieza	Blancas	Negras	
Peón	+1	-1	En un tablero se hace acumulación
Caballo	+3	-3	de valores dando un resultado positivo,
Torre	+5	-5	negativo o cero
Etc.			

[3] El concepto función de evaluación estática se toma de Winston ver [5]
[4] Véase por ejemplo Norving [3]

Los valores que aparecen +1,-1,+3,+3,+5,-5,[4] son tomados de la teoría desarrollada para este juego específico.

Aparte del valor intrínseco (el valor de cada pieza es independientemente de la posición) de las piezas se puede adicionar valores acordes con la buena posición de ellas en el tablero, el grado de seguridad de la posición, etc.

Finalmente, la función de evaluación estática deberá ser tal que a valores positivos ("máximos") indiquen que el tablero favorece a las blancas y valores negativos ("mínimos") indiquen que el tablero favorece a las negras.

Supóngase que f representa la función de evaluación estática y que la evaluación de los **tableros-hoja** en el árbol anterior, figura 4, es: f(E) =-1, f(F) = +2, f(G) = +1, f(H) = +3, f(I) = -2, f(J) = +4 representados en la figura 5.

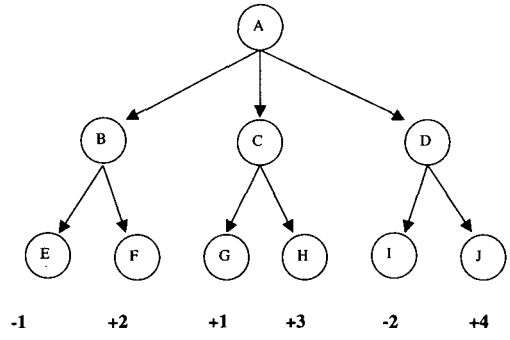

Figura 5.

Ahora las blancas reflexionan:

Si juego de modo que la posición para las negras sea B, el adversario podrá escoger entre jugadas que produzcan los tableros E o F.

El tablero F me favorece y será mi turno, pero es de suponer que el adversario juega bien, de modo que él me dejará en la situación E, y allí tendré un tablero desfavorable que defender.

Por consiguiente, "asocio con B, el **mínimo** de los valores de B. Así estaré indicando lo que sucederá si llego a B". En forma análoga se analizará C y D obteniendo una valoración subsiguiente del árbol, Figura 6.

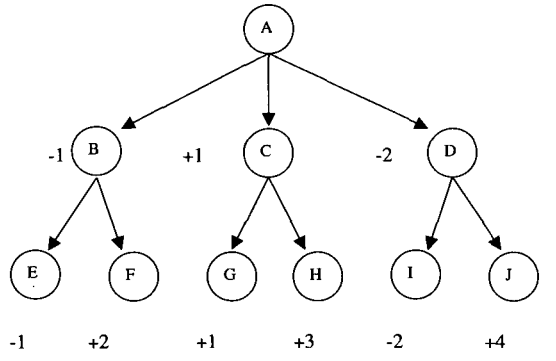

Figura 6.

Ahora el blanco establecerá lo que debe jugar:

- Si juega a B, el tablero vale -1 (favorece al adversario).
- Si juega a C, el tablero vale +1 (me favorece)
- Si juega a D, el tablero vale -2 (favorece al adversario).

Cuando es el turno de las blancas. Se toma el máximo de los valores asociados a sus hijos.

La anterior descripción del juego corresponde al procedimiento *minimax*[5]. La búsqueda *minimax* es un método para determinar movimientos. Tal búsqueda emplea una función de evaluación estática para calcular números de especificación de ventaja para las situaciones del juego, en la base de un árbol de juegos desarrollado parcialmente. Un jugador intenta obtener los números más altos, buscando la ventaja, mientras que el oponente se esfuerza por obtener los más bajos.

III. Procedimiento minimax con corte alfa-beta.

Al principio, puede parecer que la función de evaluación estática deberá utilizarse en cada nodo hoja de la base del árbol de juegos. Pero por fortuna no sucede así. Existe un procedimiento que reduce tanto el número de ramas que deben generarse como el de evaluaciones estáticas que deben hacerse, acortando así el trabajo global que se debe realizar. Es algo similar a la idea de ramificación y poda, en el sentido que algunas trayectorias son malas aún cuando no se sigan hasta el límite previsto.

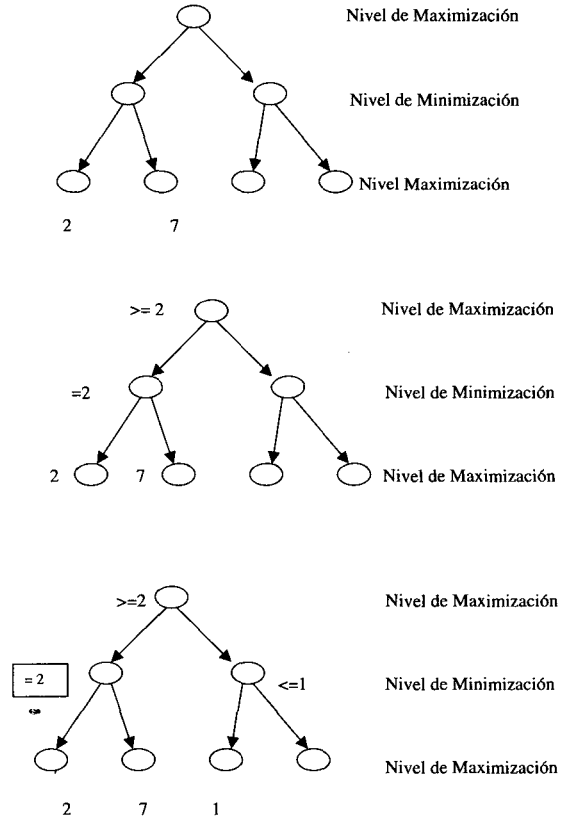

Figura 7. Procedimiento Minimax con corte alfa-beta

[5] Método descrito por Winston, ver [5]

Considere la situación que se muestra en la parte superior de la figura 7, en la cual la función de evaluación estática ya se ha utilizado en las dos primeras situaciones de los nodos hoja. El hecho de efectuar el procedimiento minimax en los resultados 2 y 7 determina que el jugador de minimización tiene garantizado un resultado de 2 si el maximizador toma la rama izquierda del nodo de la cima. Este movimiento a su vez asegura que el maximizador tiene garantizado un resultado de por lo menos 2 en la cima. Esta garantía resulta evidente aún antes de realizar otras evaluaciones estáticas, ya que el maximizador puede elegir indudablemente la rama izquierda siempre que la derecha le conduzca a un resultado más bajo. Esta situación se indica en el nodo superior de la parte media de la figura 7.

Obsérvese con más detenimiento el árbol. ¿Tiene sentido continuar hasta la situación de tablero del nodo final? ¿Acaso tiene importancia el valor producido en ese punto por la función de evaluación estática? Por extraño que parezca, la respuesta es no. Ya que indudablemente si el maximizador sabe que tiene garantizado un resultado de +2, en el caso de seguir la rama izquierda, no necesita saber más acerca de la rama derecha a parte de que en ésta no puede obtener un resultado de mayor que 1. El último nodo evaluado podría ser +100, -100 u otro valor cualquiera, sin que ello afecte el resultado. Como se muestra en la parte inferior de la figura 7, el resultado del maximizador es +2.

Al reflexionar, resulta claro que si un oponente tiene una respuesta que establece que un movimiento potencial es malo, no hay necesidad de revisar cualquiera otras respuestas ante el movimiento potencial. De forma más general, el principio alfa-beta ofrece la siguiente conclusión.

Si tiene una idea que indudablemente es mala, no se tome el tiempo para constatar qué tan mala es.

En el contexto especial de los juegos, el principio alfa-beta establece que, siempre que se descubra un hecho acerca de un nodo dado, deberá revisar lo que sabe acerca de los nodos antecesores. Puede que no resulte sensato trabajar más allá del nodo padre. Es probable, también, que lo más que Ud. llegue a esperar en el nodo padre pueda revisarse o determinarse con exactitud.

IV. Programas de Juegos

La meta es construir un algoritmo (Algoritmo de *minimax*) que a partir de una posición en que juega el computador pueda escoger entre las diversas alternativas a su disposición.

John Von Newmann y Oscar Morgenstern (1945) propusieron la idea básica de *Minimax* con la suposición fundamental de que la misma función de evaluación está disponible para ambos jugadores.

La idea de función de evaluación es perfeccionada por **Claude Shannon** (1950). **Alana Turing** en 1951 encuentra que la profundidad del árbol no necesariamente debe ser homogénea (en nuestro ejemplo, puede ser más profunda quizás para C) no será prudente evaluar en una posición envuelta en un intercambio de piezas.

En ajedrez las posiciones de jaque y capturas eventualmente originadas harán que la exploración deba continuarse hasta llegar a una posición que sea apreciada como establece para cada jugador. En este punto las posibles jugadas legales son generadas y evaluadas estáticamente.

Observará el lector en este momento, que tiene un mérito especial el traslado de valores de evaluación estática hacia arriba en el árbol, en lugar de aplicar sistemáticamente la evaluación estática en toda posición, por cuanto al desenvolverse un tablero se logran apreciar consecuencias que de otra forma no se tienen en cuenta.

V. Reporte de una investigación sobre el procedimiento minimax.

Esta sección hace parte de un trabajo de investigación que se desarrolló como tesis de maestría en Ingeniería de Sistemas en la Universidad Nacional de Colombia por el autor de este documento.

Se trabajó sobre el juego de mancala clasificado dentro de la teoría de juegos como:

- Determinístico
- De dos oponentes
- De suma cero
- Con información perfecta.

Para poder implementar eficientemente el procedimiento *minimax* con corte alfa-beta al juego mancala, fue necesario la creación de un **algoritmo iterativo,** que se trabajó a partir de los desarrollos hechos por Lauriére quien desarrolló un algoritmo iterativo para el procedimiento *minimax*, este algoritmo se corrigió y se le incorporó el procedimiento alfa-beta, el cual está disponible para la creación de juegos con las características anteriormente descritas.

A. Algoritmo minimax original.

La versión del algoritmo en Francés es la siguiente:

```
MINIMAX :    DONNEES: (echiquier, camp)
```

```
            RESULTAT : (valeur de la position = minimax)
            Prof = 1 ; camp =1 ; E(1) = pilocoups (echiquier, camp);
            Eval(1) = -oo

REPETER TANT QUE  porf > 1
      REPETER TANT QUE E(prof) ¡= vide
            Copu  = sommet – pile (E(prof))
            Echiquier = jouer (echiquier, coup)
            Si prof ¡= (profmax – 1)
               ALORS  prof = prof +1
                      Camp =  - camp
                      E (prof) = pilecoups (echiquier, campo)
                       eval(prof) = - oo
               SINON   {profmax: evaluer la position et comparer}
                      Eval (prof) = MAX [eval(prof),camp*valuation (echl)]
                       {dejouer le dernier coup a profmax:}
                       coup = depile(E(prof))
                       echiquier = dejouer (echiquier, coup)
                        {rester a cette proffondeur tant qu'll reste des coups}
            FSI
      FR sur E
   {retour arriere}
   SI prof=1 ALORS minimax = eval(1) :fin FSI
   Camp = - camp
   Prof   = prof-1
   {Dejouer le dernier coup a cette profondeur }
    coup = depile ( E (prof))
    echiquier = dejouere (echiquier,copu)
    { Remonter L'evaluation par minimax}
     eval (prof) = MAX [eval (prof) – ( eval (prof +1))]
FR sur prof
FIN minimax.
```

B. ALGORITMO MODIFICADO

El siguiente algoritmo tiene las modificaciones necesarias hechas sobre el algoritmo iterativo propuesto por Lauriére para que opere con corte alfa-beta, estas modificaciones son aportes del autor en su tesis de maestría.

Entrada : (tablero,lado)
Salida : Valor del la Posición.
Inicializacion: prof1; E(1) = listadejugadas(tablero,lado);Eval(1)=-(lado)*alfa
(tomamos alfa como el signo para infinito)

```
      mientras  prof >=1
       hacer
       mientras E(prof) <> vacio
              hacer
              jugada = primera (E(prof);
              tablero = Jugada(tablero,jugada);
              si prof <> (profmaxima –1);
              entonces
                     hacer
                     prof = prof +1;
                     lado = -lado;
                     E(prof) = lista de jugadas(tablero,lado);
                     Eval(prof) = -(lado) * alfa;
                     Fin_hacer
```

```
Sino
        Hacer
        Sw=1;
        Si lado=1
        Entonces
                Hacer
                        Eval(prof)=  MAX[eval(prof),valor(tablero)];
                        Si prof >1
                        Entonces
                                hacer
                                si eval(prof) >= eval(prof-1)
                                entonces
                                        hacer
                                        eliminar(E(prof));
                                        SW=0;
                                        Fin_hacer
                                Fin_hacer
                Fin_hacer
        Sino
                Hacer
                Eval(prof) = MIN[eval(prof),valor(tablero)]
                Si prof > 1
                Entonces
                        Hacer
                        Si eval(prof) <= eval(prof-1)
                        Entonces
                                Hacer
                                Eliminar(E(prof));
                                Tablero = restaurar(tablero,jugada);
                                Fin_hacer
                        Fin_hacer
                Fin_hacer
        Si SW=1
        Entonces
                Hacer
                Jugada = sacarprimera(E(prof));
                Tablero = restaurar(tablero,jugada);
                Fin_hacer
        Fin_hacer
Fin_hacer

Si prof = 1
Entonces
        Hacer
        Retornar(eval(1))
        Terminar
        Fin_hacer
Lado = -lado;
Prof  = prof –1;
Jugada = sacarprimera(E(prof));
Tablero = restaurar(tablero,jugada);
Si lado=1
Entonces
        Hacer
        Eval(prof) = MAX[eval(prof),eval(prof + 1)];
        Si prof>1
        Entonces
```

```
                Si eval(prof) >= eval(prof-1)
                Si eval(prof) ¡= vacio()
                Entonces eliminar(E(prof));
        Fin_hacer
Sino
        Hacer
        Eval(prof) = MIN[eval(prof),eval(prof + 1)];
        Si prof >1
        Entonces
                Si eval(prof) <= eval(prof-1)
                Si eval(prof) ¡= vacia()
                Entonces eliminar(E(prof));
        Fin_hacer
        Fin_hacer
```

C. Aportes al algoritmo

Los aporte dados al algoritmo[6] consistieron en:

- El algoritmo en sus versiones conocidas, presentados por Shapiro y Lauriére traían errores así como: SI PROF <> (PROFMAX) esta parte debe quedar como : SI PROF <> (PROF MAX –1). (Si profundidad es diferente a profundidad máxima menos uno), para que se tome en consideración el nivel máximo menos uno.

- Se introdujo el switch SW, para controlar el corte alfa/beta si SW= 1 permite sacar el primer elemento de E(prof) y guardarlo en el nodo jugada. Además permite restaurar el tablero reemplazándolo por el tablero almacenado en jugada y colocarlo nuevamente en tablero. Si SW =0 no permite este procedimiento. En otras palabras Tenemos:

```
si SW =1 entonces
        jugada = sacarprimera(E(prof));
                tablero = restaurar(tablero,jugada);
                fin_si.
```

- Se modificó la siguiente parte del algoritmo para obtener el corte alfa -beta.

```
Else [maxdepth:evaluate the position and compare]
Eval(depth) = MAX[eval(depth.side,value (board)]
Move = unstack(E(depth)) [cancel previous move]
Boar = restore(board,move) [remain at this depth so long as there are moves]
```

La nueva versión con su modificación corte alfa -beta es.

```
Sino
    Hacer
    Sw=1;
    Si lado=1
    Entonces
            Hacer
                    Eval(prof)=  MAX[eval(prof),valor(tablero)];
                    Si prof >1
                    Entonces
                            hacer
                            si eval(prof) >= eval(prof-1)
                            entonces
                                    hacer
                                    eliminar(E(prof));
                                    SW=0;
                                    Fin_hacer
                            Fin_hacer
            Fin_hacer
    Sino
            Hacer
            Eval(prof) = MIN[eval(prof),valor(tablero)]
```

[6] Los aportes al algoritmo fueron tomados de la tesis de maestría, aplicación de estrategias de búsqueda en el juego de mancala véase [1].

```
                    Si prof > 1
                    Entonces
                        Hacer
                        Si eval(prof) <= eval(prof-1)
                        Entonces
                            Hacer
                            Eliminar(E(prof));
                            Tablero = restaurar(tablero,jugada);
                            Fin_hacer
                        Fin_hacer
                    Fin_hacer
                Si SW=1
                Entonces
                    Hacer
                    Jugada = sacarprimera(E(prof));
                    Tablero = restaurar(tablero,jugada);
                    Fin_hacer
                Fin_hacer
            Fin_hacer
```

Otra parte del algoritmo modificada es la que tiene que ver con:

```
if depth –1 then minimax <- eval(1) endif
side <- -side
depth <- depth -1
move <- unstack(E(depth)) {cancel last move at this depth}
board <- restore(board,move)
eval(depth) <- MAX{eval(depth), - eval(depth + 1)} {minimax evaluation}
end repeat detpth
```

La nueva versión del algoritmo que hacer el corte alfa-beta quedó estructurado como sigue:

```
Si prof = 1
    Entonces
        Hacer
        Retornar(eval(1))
        Terminar
        Fin_hacer
    Lado = -lado;
    Prof  = prof –1;
    Jugada = sacarprimera(E(prof));
    Tablero = restaurar(tablero,jugada);
    Si lado=1
    Entonces
        Hacer
        Eval(prof) = MAX[eval(prof),eval(prof + 1)];
        Si prof>1
        Entonces
            Si eval(prof) >= eval(prof-1)
            Si eval(prof) ¡= vacio()
            Entonces eliminar(E(prof));
        Fin_hacer

    Sino
```

```
Hacer
Eval(prof) = MIN[eval(prof),eval(prof + 1)];
Si prof >1
Entonces
        Si eval(prof) <= eval(prof-1)
        Si eval(prof) ¡= vacia()
        Entonces eliminar(E(prof));
Fin_hacer
Fin_hacer
```

Los algoritmos expuestos en los libros casi siempre son demasiado generales, e implican para su implementación un trabajo dispendioso. El desarrollo del anterior algoritmo iterativo para el procedimiento minimax con corte alfa-beta, es básico y está disponible para cualquier persona interesada en desarrollar juegos clasificados dentro de la teoría como determínistico, de suma cero, de dos oponentes y con información perfecta.

Sin embargo, hasta el momento no se comprende suficientemente cómo combinar los algoritmos de búsqueda heurística (búsqueda preferente por lo mejor, búsqueda limitada por la capacidad de memoria y algoritmos de mejoramiento iterativo), y los algoritmos de juegos (minimax, minimax con corte alfa-beta) en un sólo sistema robusto que permita un mejor aprovechamiento de las técnicas de Inteligencia Artificial. Un algoritmo que permita esta unificación será una herramienta de gran ayuda para el tratamiento de juegos.

Bibliografía

1.BECERRA CORREA, Nelson. ***Aplicación de estrategias de búsqueda en el juego de mancala***, Tesis de maestría, Facultad de Ingeniería, Universidad Nacional de Colombia. 1998.

2.CAIRO, Osvaldo y GUARDATI, Silva. ***Estructuras de Datos***. McGraw-Hill, Mexico. 1993.

3.RUSELL, Stuart y NORVING, Peter. ***Inteligencia Artificial. Un enfoque moderno.*** Prentice Hall. 1996.

4.VON NEWMANN, Jhon. ***Collected works***. Ed. A. H Taub-Oxford: Pergamon Press. 1.976.

5.WINSTON, Patrick Henry. ***Inteligencia Artificial.*** Addison-Wesley Iberoamericana. 1.996.