Ingeniería en computación
Universidad de La Serena
II Semestre, 2020
Versión número 1

Modelo Solución

AIPA

Lunes 23 de noviembre

Integrantes:
Yair Gallardo
Norton Irarrázabal
Sebastian Rojas

Docente
Guillermo Leyton.

Propósito

Este documento tiene como finalidad generar un modelo solución que satisfaga lo especificado en la meta. Este modelo debe ser explicativo, conciso y sin ambigüedades. Debe considerar la realidad en donde se desenvuelve y ser apoyado por diagramas que logren clarificar el desarrollo de la solución, en beneficio de la implementación del software.
Está dirigido a todo aquel que se interese por el desarrollo del software AIPA o bien por la problemática en la que se encuentra inmerso.

Descripción del problema

El uso de inteligencia artificial en el ámbito de los jugadores virtuales, es una **herramienta de uso investigativo y publicitario** que cuenta con larga data dentro de los últimos 30 años. En el área investigativa, se pretende encontrar el alcance de las capacidades de simulación racional con las que cuentan las máquinas; el logro de altos niveles por estos jugadores virtuales suele generar gran interés en el público.

Otra perspectiva a valorar es el amplio mercado de los videojuegos, donde la dificultad, la búsqueda de retos y experiencias satisfactorias suelen ser una de las áreas de interés para los jugadores más exigentes.
Intereses que pueden ser abordados por estos jugadores virtuales, siendo un reto y por otra una “ayuda” para jugadores menos experimentados que busquen encontrar o descubrir nuevas jugadas que faciliten su éxito mejorando su experiencia y satisfacción. Por consiguiente es de interés para empresas desarrolladoras que buscan satisfacer a este tipo de clientes.

Si bien el software a desarrollar se desenvolverá en un ámbito concreto, juego de estrategia **ajedrez**; el enfoque es el mismo, siendo un buen prototipo aplicable en otros entornos.

Propuesta

Para abordar el problema se plantea el uso de agentes inteligentes, partiendo del siguiente análisis.

REAS

Tipo de agente	Medidas de rendimiento	Entorno	Actuadores	Sensores
Jugador de ajedrez	● Balance de materiales. ● Movilidad. ● Control del tablero. ● Conectividad.	● Tablero	● Mover una pieza. ● Capturar una pieza. ● Realizar enroques. ● Realizar “al paso”. ● Promover una pieza.	● Localización de las piezas en el tablero.

Jugador de ajedrez guia	• Balance de materiales. • Movilidad. • Control del tablero. • Conectividad.	• Tablero	• Realizar recomendaciones	• Localización de las piezas en el tablero.

Balance de materiales.
Se refiere a que cada pieza del tablero tiene un valor determinado:

Peón: 200, Caballo: 320, Alfil: 330, Torre: 500, Reina: 900, Rey: 20000 (Es el estándar)

Source	Year	Pawn	Knight	Bishop	Rook	Queen
H. S. M. Coxeter [7]	1940		300	350	550	1000
Max Euwe and Hans Kramer [8]	1944	100	350	350	550	1000
Claude Shannon [9]	1949	100	300	300	500	900
Alan Turing [10]	1953	100	300	350	500	1000
Mac Hack [11]	1967	100	325	350	500	975
Chess 4.5 [12]	1977	100	325	350	500	900
Tomasz Michniewski [13]	1995	100	320	330	500	900
Hans Berliner [14] [15]	1999	100	320	333	510	880
Larry Kaufman [16]	1999	100	325	325	500	975
Fruit and others [17]	2005	100	400	400	600	1200
Larry Kaufman [18]	2012	100	350	350	525	1000

En los libros de ajedrez también se puede encontrar como:

- Peón: 1.
- Caballo o alfil: 3
- Torre:5
- Reina 9.

En donde el balance se refiere a la suma de las piezas del jugador menos las del contrincante.

Movilidad.
Se refiere a la cantidad de cuadros que pueden alcanzar cada una de las piezas en un solo movimiento, el cual va a depender de la localización en donde se encuentren cada una de las piezas.

Control del tablero.
Se refiere a la cantidad de cuadros vacíos que abarcan las piezas del jugador en el tablero.

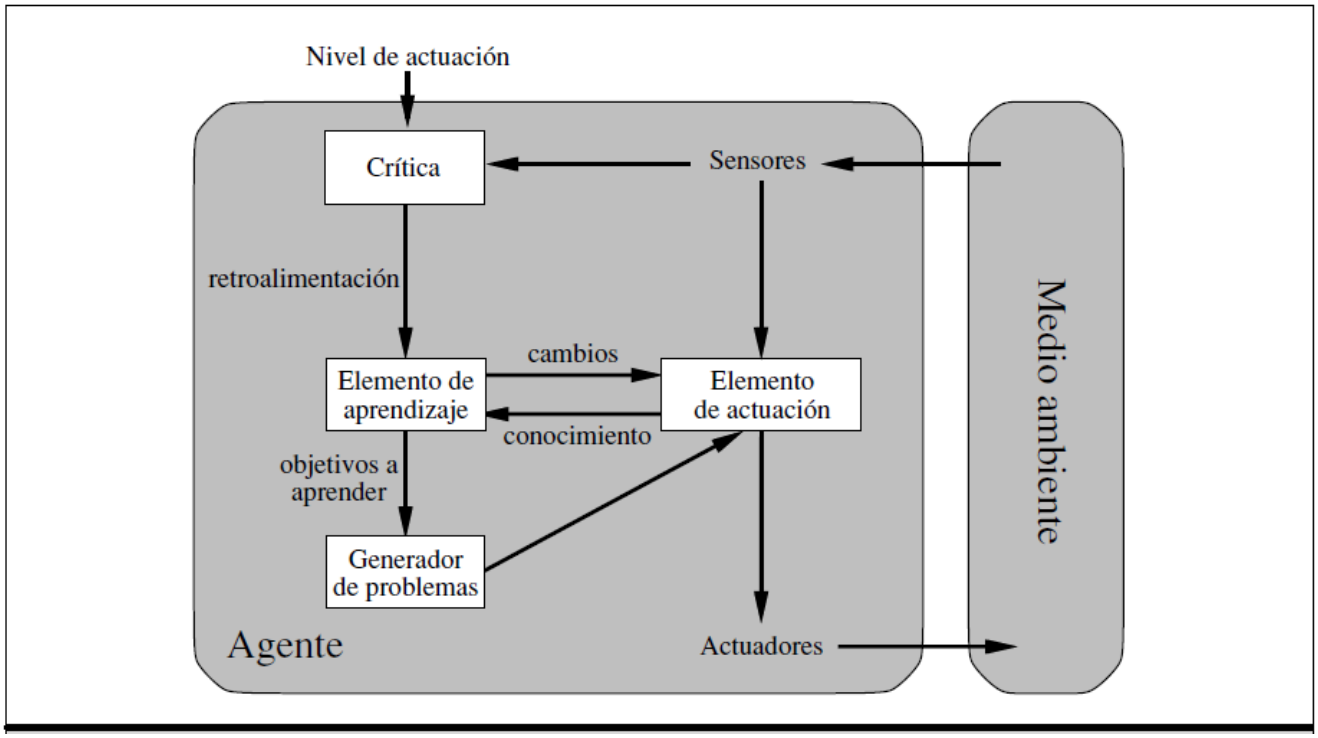
Conectividad.
Indica el grado de conectividad entre las piezas, es decir la pieza A por cuentas piezas está siendo protegida.

Propiedades del entorno

Las propiedades del entorno son las siguientes:

- **Totalmente observable**: Se tiene acceso al estado completo del medio, de todos los aspectos relevantes en la toma de decisiones.
- **Estratégico**: El medio es determinista pero solo para sus propias acciones no para las acciones del otro agente.
- **Secuencial**: Las decisiones del presente afectarán a las decisiones del futuro.
- **Estático**: El agente no se preocupa por el paso del tiempo mientras está deliberando ya que el entorno no cambia hasta que realiza su jugada.
- **Discreto**: Posee un número finito de estados, un conjunto discreto de percepciones y acciones.
- **Multiagente competitivo**: El juego de ajedrez está en un entorno de dos agentes en donde la entidad oponente B intenta maximizar su medida de rendimiento,la cual minimiza la medida de rendimiento del agente A.

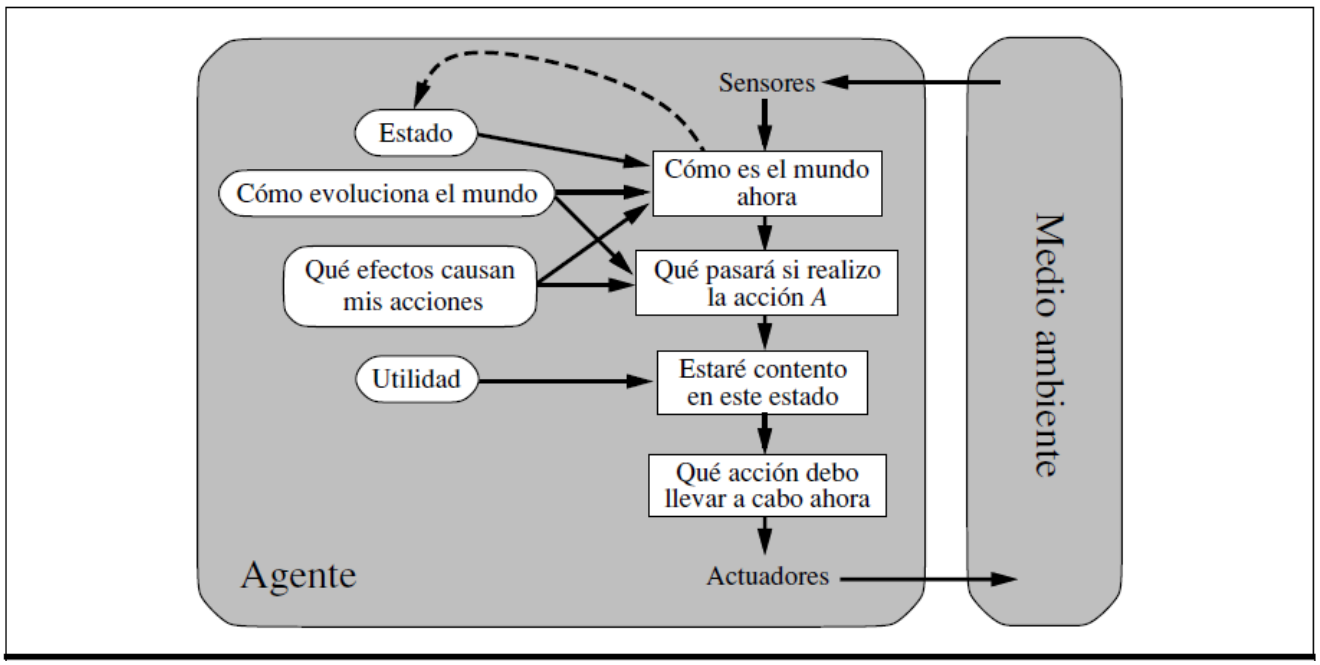
Modelo general para agentes que aprenden.



El cual se divide en dos partes el elemento de aprendizaje y el elemento de actuación:

- **Elemento de aprendizaje**: Responsabilizado de hacer mejoras.
    - El elemento de aprendizaje se realimenta con las **críticas** de la actuación del agente.
- **Elemento de actuación**: Responsabilizado de selecciones de acciones.
    - El elemento de actuación es el equivalente a un agente completo, en este caso el que más se asemeja a lo que buscamos es el agente basado en utilidad ya que el agente basado en meta es insuficiente para lo que requerimos ya que solo proporciona una distinción binaria entre alcanzó y no alcanzó; 1 y 0.
- **Generador de problemas**: Responsable de sugerir acciones que lo guiaran a experiencias nuevas e informativas.

Modelo general para agentes basados en utilidad.



Búsqueda

Al ser complicado llegar a un estado objetivo se plantea realizar **Búsqueda,** la cual es la encargada de encontrar secuencias de acciones que permiten al agente alcanzar su meta (a esto se le conoce como agente resolvente de problemas), en este caso llegar al estado jaque mate, sin embargo explorar todas las opciones resulta en un árbol muy grande, por lo que se añade la función de evaluación la cual considerará los criterios especificados en medidas de rendimiento para reducir el espacio de búsqueda, implicando la poda de nodos y subárboles cuando se estima que sus valores no afectarán a la decisión final, con esto nos referimos a lo siguiente:

- Cuando diferentes rutas conducen a un estado idéntico respecto a un rendimiento.
- Cuando las otras rutas tienen un menor rendimiento frente a una ruta ya explorada.

El agente inteligente adicionalmente debe decidir si efectuar acciones de explotación es decir tomar acciones que conduzcan a una alta recompensa o tomar acciones de exploración es decir nuevas acciones que podrían conducir a una mayor recompensa.

Por otra parte se debe tener en cuenta que una acción a corto plazo no sea óptima pero si a largo plazo.

El tipo de búsqueda que describe este problema es la búsqueda adversaria la cual está presente en multiagentes competitivos en donde los objetivos se oponen entre sí.

Dónde la búsqueda se basa en la construcción de una estructura llamada árbol de juego el cual es un grafo dirigido:

- En donde los nodos representan estados es decir un patrón generado por las posiciones de las piezas en el tablero.
- Sus vértices representan movimientos.
- Los nodos hojas representan un estado final.

Bajo este contexto de entorno multiagente competitivo, en los cuales el agente tendrá que considerar acciones de otros agentes y cómo afectan a su propio bienestar.

A esto se le denomina búsqueda adversario, a los que los teóricos lo llaman juegos de suma cero, ya que los valores de utilidad al final del juego son siempre iguales y opuestos. Por ejemplo si un jugador gana el juego (+1), por lo tanto el otro jugador necesariamente pierde (-1). Por lo que esta oposición entre las funciones de utilidad de los agentes hace la situación entre adversarios.

En este contexto el ajedrez tiene un factor de ramificación promedio de 35. Por lo que el árbol de búsqueda tiene aproximadamente 35^{100}nodos. Por lo tanto se requiere la capacidad de tomar alguna decisión cuando es factible calcular la decisión óptima.

En ese sentido se plantea la poda que nos va a permitir ignorar partes del árbol que no marcan ninguna diferencia para obtener la opción final, y las funciones de evaluación heurísticas nos permitirán aproximar la utilidad verdadera de un estado sin hacer una búsqueda completa.

El juego puede definirse formalmente como una clase de problemas de búsqueda con los siguientes componentes:

- El estado inicial, que incluye la posición del tablero e identifica al jugador que mueve.
- Una función sucesor, que devuelve una lista de pares (movimiento, estado), indicando un movimiento legal y el estado que resulta.
- Un test terminal, que determina cuando se termina el juego. A los estados donde el juego se ha terminado se les llama estados terminales.
- Una función de utilidad, que da un valor a los estados terminales. Triunfo +1, empate 0, perdida -1.

El estado inicial y los movimientos legales para cada lado (para cada jugador) definen el árbol de juegos.

Ahora se plantea que tenemos 2 jugadores a uno de ellos se le llamará MAX y al otro MIN. Entonces se tiene el árbol de juegos en el que sobre las hojas existe un número que indica la utilidad del estado terminal desde el punto de vista de MAX los valores altos son buenos y malos para MIN.

Para realizar una estrategia óptima se debe seguir una secuencia de movimientos que conducen a un estado objetivo, un estado terminal que es ganador.

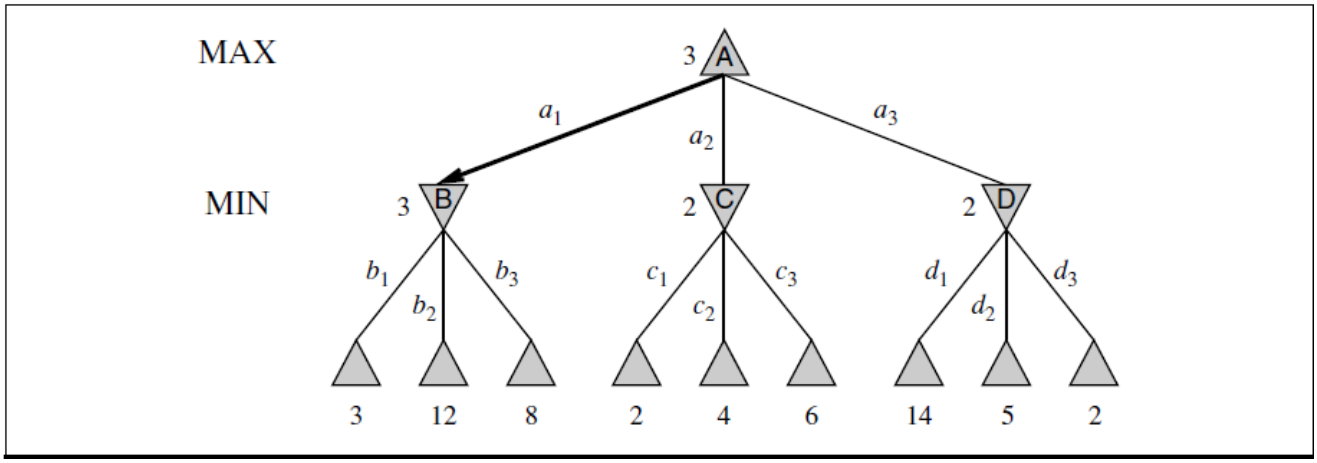
A cada movimiento realizado en el árbol le denominamos capa (profundidad).
Considerando el árbol de juegos, la estrategia óptima puede determinarse examinando el valor minimax de cada nodo (valor-minimax(n)).

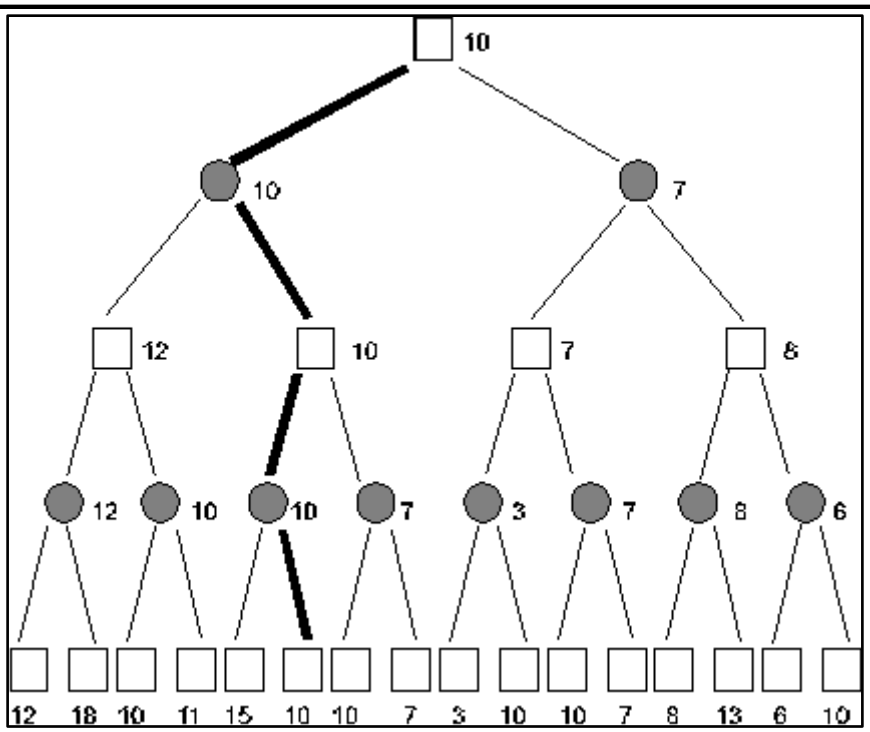
valor-minimax(n)=

- utilidad(n) si n es un estado terminal.
- $max_{s \in Sucesores}$valor-minimax(s) si n es un estado MAX.

- $min_{s \in Sucesores}$valor-minimax(s) si n es un estado MIN.

Ejemplo:



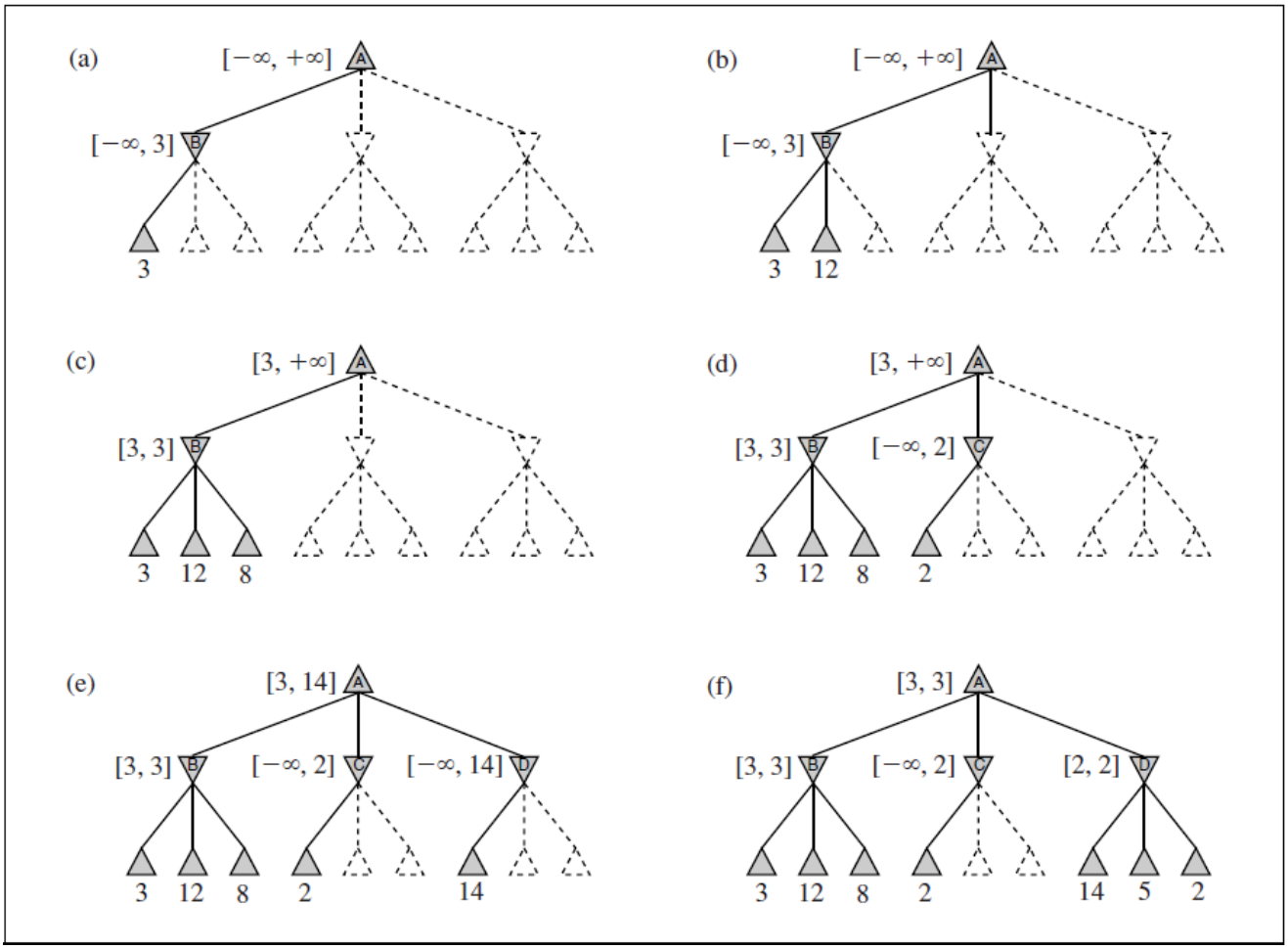


El algoritmo minimax usa un cálculo simple recurrente de los valores minimax de cada estado sucesor. La recursión avanza hacia las hojas del árbol , y entonces los valores minimax retroceden por el árbol cuando la recursión se va deshaciendo.

Este algoritmo realiza exploración “primero en profundidad completa” del árbol de juegos. Si la profundidad del árbol es “m” y hay “b” movimientos legales. Entonces la complejidad en tiempo del algoritmo minimax es O(b^m). La complejidad en espacio es O(b*m).

Convirtiendo este algoritmo demasiado ineficiente, ya que el problema es el número de estados que se tienen que examinar resulta exponencial. Por lo que se plantea dividir este valor, evitando mirar todos los nodos en el árbol de juego. Eliminando partes grandes del árbol.

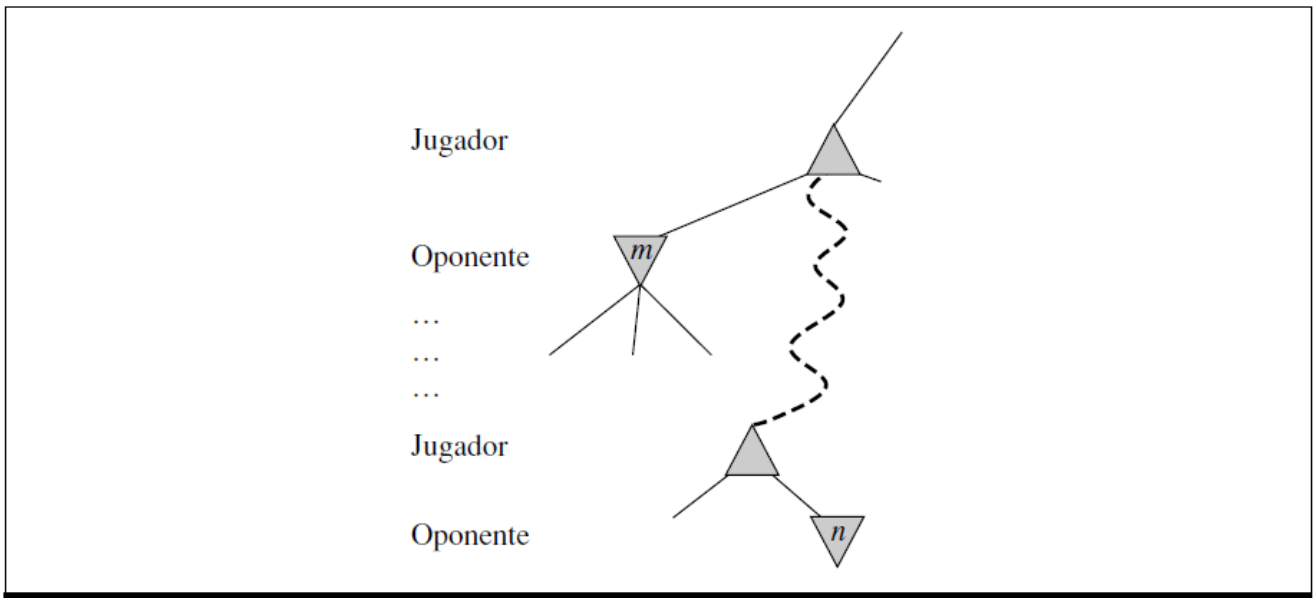
Poda alfa-beta



Nodos no evaluados del nodo C = x e y, sea z el mínimo entre x e y.

$$MINIMAX - VALUE\ (raíz) = max(min(\mathbf{3},\mathbf{12},\mathbf{8}), min(\mathbf{2}, x, y), min(\mathbf{14},\mathbf{5},\mathbf{2}))$$
$$= max(\mathbf{3}, min(\mathbf{2}, x, y), \mathbf{2})$$
$$= max(\mathbf{3}, z, \mathbf{2})\ donde\ z <= \mathbf{2}$$
$$= \mathbf{3}$$

Es decir el valor de la raíz y la decisión son independientes de los valores de las hojas podadas x e y.
La poda alfa-beta puede aplicarse a árboles de cualquier profundidad, y a menudo es posible que sea posible podar subárboles enteros.



En el caso general, Si m es mejor que n para el jugador, nunca iremos a n en el juego.

La poda alfa-beta tiene nombre debido a dos parámetros que describen los límites sobre los valores hacia atrás que aparecen a lo largo del camino.

- α = El valor de la mejor opción (es decir, el más alto) que hemos encontrado hasta ahora en cualquier punto elegido a lo largo del camino para MAX.
- β =El valor de la mejor opción (es, decir el valor más bajo) que hemos encontrado hasta ahora en cualquier punto elegido elegido a lo largo del camino para MIN.

La búsqueda alfa-beta actualiza el valor de α y βsegún se va recorriendo el árbol y poda las ramas restantes de un nodo (es decir, termina la llamada recurrente) tan pronto como el valor del nodo actual es peor que el actual valor α o βpara MAX o MIN.

La eficacia de la poda alfa-beta es muy dependiente del orden en que se examinan los sucesores, esto sugiere que primero debemos examinar los nodos sucesores que probablemente sean los mejores. De esta forma se reduce la cantidad de nodos a examinar para escoger el mejor movimiento a O($b^{m/2}$). Reduciendo el factor de ramificación aproximadamente de 6 a 35 en el ajedrez.

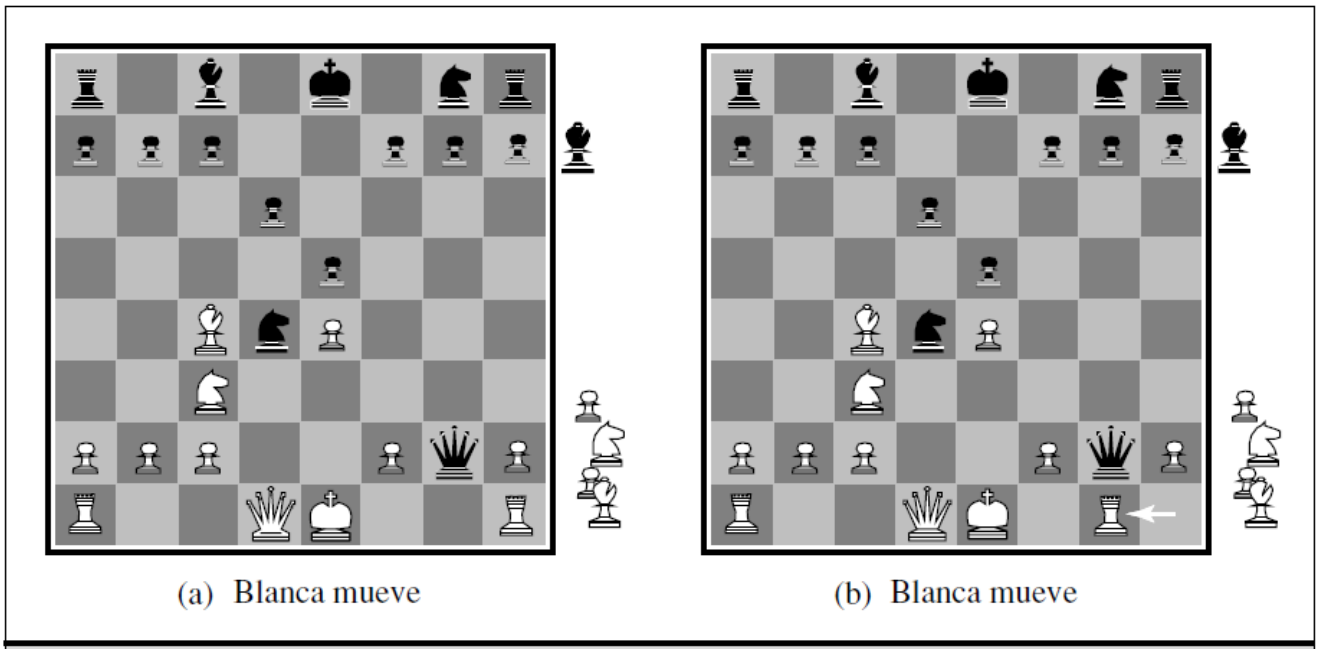
Esto puede ser abordado a través de una función de ordenación sencilla como intentar primero las capturas, luego las amenazas, luego mover hacia adelante, y por último los movimientos hacia atrás.

El algoritmo minimax genera el espacio de búsqueda entero, mientras que el algoritmo alfa-beta permite que podemos partes grandes de él. Sin embargo, alfa-beta todavía tiene que buscar en los caminos, hasta los estados terminales, para una parte del espacio de búsqueda, lo que es poco práctico ya que los movimientos deben hacerse en una cantidad razonable de tiempo.
Por lo tanto se propone un cambio de transformar los nodos no terminales en nodos terminales. Sustituyendo la función de utilidad por una función de evaluación heurística EVAL, que da una estimación de la utilidad de la posición y se sustituye el test-terminal por un test límite que decide cuándo aplicar EVAL.

Función de evaluación

La función de evaluación devuelve una estimación de la utilidad esperada de una posición dada.

Durante siglos los jugadores de ajedrez han desarrollado modos de juzgar el valor de una posición. Por lo tanto el funcionamiento de un programa de juegos es dependiente de la calidad de su función de evaluación. Ya que una función de evaluación inexacta llevará al agente hacía posiciones que resultan estar perdidas. Por otra parte está función debe tener la restricción de no utilizar demasiado tiempo.



(a) Blanca mueve (b) Blanca mueve

La calidad de la función de evaluación depende de la selección de caracteristicas, ya que podemos ver que en el caso a) el negro tiene una ventaja de un caballo y dos peones lo que se podria considerar como una victoria para este, sin embargo si el blanco se come a la reina genera totalmente el caso opuesto.

Adicionalmente la función de evaluación debe estar fuertemente correlacionada con estados terminales que permitan ganar.

La mayoría de las funciones de evaluación trabajan calculando características del estado. Algunos conjuntos de características conducen a victorias, empates o pérdidas.
La función de evaluación calcula la contribución numérica de cada característica y luego las combinan para encontrar el valor total de la posición. (lo especificado al inicio)

$$\text{EVAL}(s) = w_1 f_1(s) + w_2 f_2(s) + \cdots + w_n f_n(s) = \sum_{i=1}^{n} w_i f_i(s),$$

- wi: pesos.
- fi: característica de la posición.

Sin embargo esta función es una función lineal, el problema es que cada característica es independiente de los valores de otras características, Por lo que lo recomendable es utilizar funciones no lineales de características.

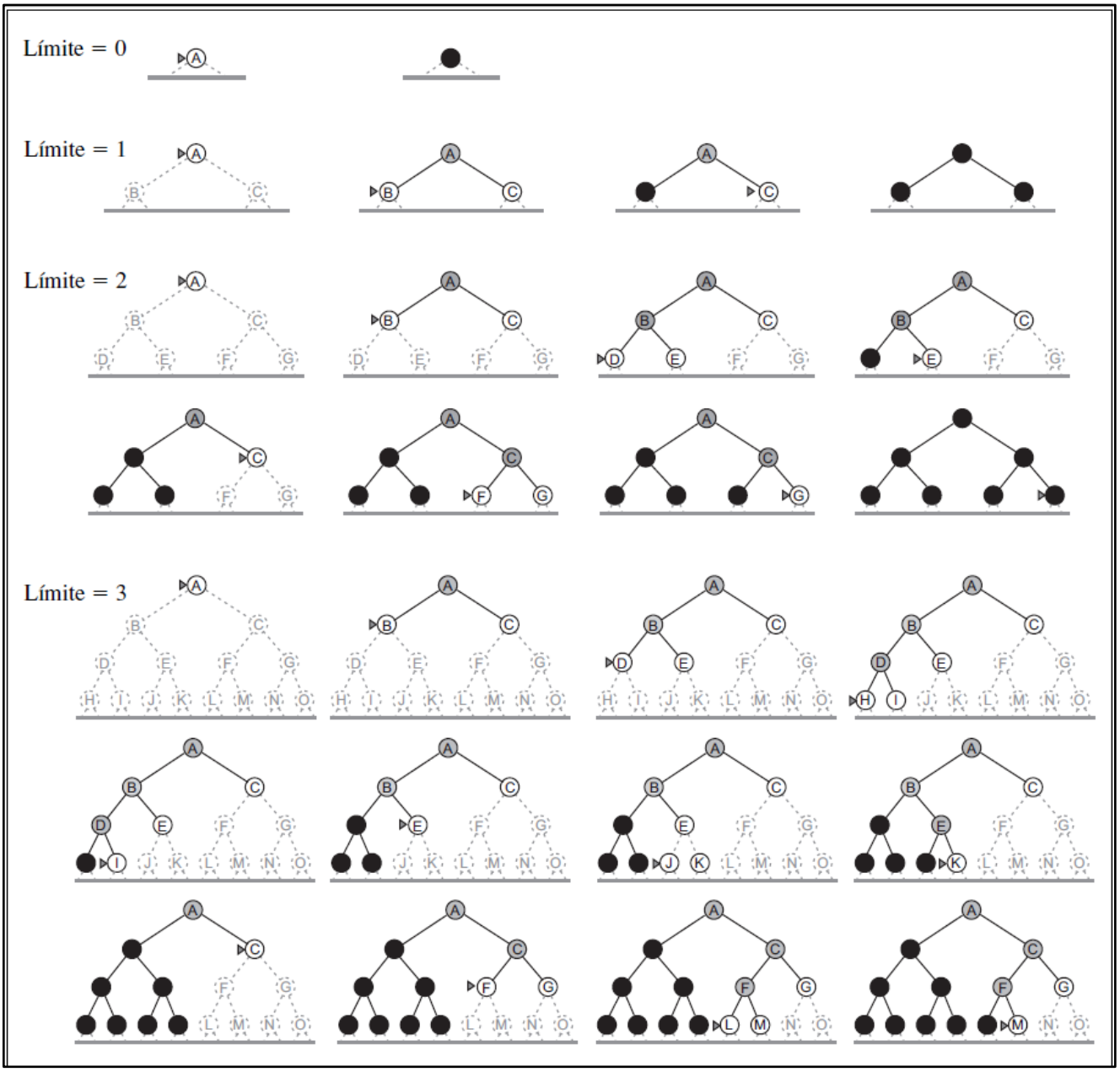
Por ejemplo: Un par de alfiles podría merecer más la pena (es decir una mayor ponderación en conjunto) que dos veces el valor de un alfil (fijo), y un alfil merece más la pena en la fase final que al principio.

Corte de la búsqueda

El siguiente paso es modificar la búsqueda-alfa-beta de modo que llame a la función heurística EVAL cuando se corte la búsqueda reemplazando el test-terminal por test-corte.
Llevando la contabilidad de la profundidad de modo que la profundidad actual se incremente sobre cada llamada recursiva.

La forma más sencilla de controlar la búsqueda es poner un límite de profundidad fija, de modo que el test-corte devuelva verdadero para toda profundidad mayor que alguna profundidad d. La profundidad d se elige de modo que la cantidad de tiempo usado no exceda lo que permiten las reglas del juego.

Una aproximación a esto es aplicar profundidad iterativa. Sin embargo, la función de evaluación debe aplicarse solo a posiciones que son improbablemente expuestas a grandes oscilaciones en su valor futuro próximo. Se debe buscar la estabilidad.



Extensiones o adiciones al modelo

Para solucionar este problema se plantea utilizar aprendizaje por refuerzo basado en modelo, para calcular los Q-valor mediante frecuencia relativa es decir el agente juega toma una serie de decisiones (también se le conoce como políticas) en cada estado y esa secuencia de acciones nos llevó a una recompensa final que puede ser ganar, perder o empatar en base a eso el agente actualiza los valores propaganadolo de abajo hacía arriba de las hojas hacia la raíz +1, evidentemente los estados más cercanos a los terminales deben ver afectado su valor en mayor medida que aquellos nodos más lejanos ya que tienen una implicación mayor en el resultado final.