Universidad de las Ciencias Informáticas
Facultad 5

Título: Juego de ajedrez interactivo como medio de enseñanza.

Trabajo de Diploma para optar por el título de

Ingeniero en Ciencias Informáticas

Autores: Yordany Ricardo Marrero

Gustavo López Rivera

Tutor: MSc. Roberto Millet Luaces

Co-tutor: MSc. Irisbel Fuentes Benítez

Junio del 2009

Masificar el ajedrez colocaría a este país con mucha más capacidad de pensar, más eficiente; es como saber una asignatura básica.

Fidel Castro

DATOS DE CONTACTO

Tutor: Roberto Millet Luaces

Breve currículo:

- Graduado de Ingeniero Eléctrico en 1986, en la Universidad de Camagüey.
- Profesor Auxiliar.
- MSc. en Ciencias Matemáticas.
- Imparte docencia en universidades desde 1987.

Ubicación: Universidad de las Ciencias Informáticas, Cuba.

E-mail: milletp@uci.cu

Co-tutor: Irisbel Fuentes Benítez

Breve currículo:

- Licenciado en Cultura Física en 2004, en la Facultad de Cultura Física de Pinar del Río.
- Profesor Asistente.
- MSc. en Actividad Física en la Comunidad.
- Asignado al proyecto Futuro en el 2004 por ser graduado más destacado en la Investigación.

Ubicación: Universidad de las Ciencias Informáticas, Cuba.

E-mail: irisbelfb@uci.cu

AGRADECIMIENTOS

A Fidel Castro y la Revolución por brindarnos la posibilidad de ser alguien en la vida.

A la Universidad de las Ciencias Informáticas por forjarnos como profesionales y personas de bien.

A nuestras familias por brindarnos toda su fuerza, amor y apoyo incondicional.

A nuestra otra familia de la UCI por estar siempre presente cuando la necesitamos, por brindarnos su apoyo, por compartir buenos y malos momentos con nosotros, en fin, muchas gracias por ser una parte importante en nuestras vidas.

A nuestro tutor y cotutor, por tener tanta paciencia y tratar de hacer de nosotros excelentes profesionales.

Al Ingeniero Raciel por las tantas ideas buenas que nos brindó.

Y a aquellas otras personas que de una forma u otra nos ayudaron en la realización de este trabajo. A todos una vez más……..MUCHAS GRACIAS.

DEDICATORIA

A mis padres, por tanto amor y confianza depositados en mí, por enseñarme a luchar por mis sueños y brindarme su apoyo en todos los momentos de mi vida y más aún durante la realización de esta investigación.

A mis hermanas por servirme como ejemplo para llegar hasta donde estoy.

A mi compañero de tesis, que es además uno de mis mejores amigos, gracias por estar ahí en todos estos momentos difíciles, siempre supimos conformar un buen equipo.

A mi novia por su especial apoyo durante todo este tiempo, por ser esa persona especial que has sido para mí, gracias.

Gustavo

A mi familia, por la entera confianza, el cariño, el amor y el apoyo ofrecido de todo corazón.

A mi abuelo Arsenio, por los valores humanos inculcados que me han convertido en una persona de bien.

A mi compañero de tesis, por ser además, un buen amigo que ha compartido conmigo buenos y malos momentos de la vida universitaria.

Yordany

RESUMEN

En la Universidad de las Ciencias Informáticas se imparte un curso para la enseñanza del ajedrez como parte de la asignatura de Educación Física. Existe una plataforma de aprendizaje donde se evalúa al estudiante mediante cuestionarios que contienen preguntas e imágenes de tableros de ajedrez con posiciones donde el estudiante debe seleccionar de varios movimientos descritos el correcto. Dicha plataforma no cuenta con una aplicación donde el estudiante pueda aprender a jugar ajedrez de forma interactiva.

El objetivo de este trabajo es desarrollar un juego de ajedrez interactivo que le permita al estudiante poner a prueba sus conocimientos y desarrollar habilidades en el juego ciencia. Para darle solución al problema se utilizan técnicas de Inteligencia Artificial mediante las cuales se pueden analizar las jugadas realizadas por el estudiante para brindarle consejos, sugerencias y advertencias sobre la posición del tablero en cada uno de sus movimientos. También se aplican técnicas de Inteligencia Artificial que le permite al programa de ajedrez jugar contra el usuario después que este haya configurado el tablero de juego a su conveniencia. Para ello se hace uso del programa o motor de ajedrez GNU Chess, al que se le realizan modificaciones para lograr que el usuario se guíe paso por paso de cada uno de los movimientos del juego que va llevando a cabo en cuanto a conceptos, ventajas, inconvenientes, tácticas y estrategias que debe tener muy en cuenta para lograr un buen resultado.

PALABRAS CLAVE

Ajedrez, GNU Chess, Inteligencia Artificial

ÍNDICE

INTRODUCCIÓN

El acelerado avance del razonamiento humano ha propiciado el surgimiento de múltiples ciencias y ramas que cada día con el uso de las nuevas tecnologías se perfeccionan más. Dentro de este inmenso campo de sabiduría conformado a lo largo de la historia por mentes brillantes, surgió el ajedrez. Este juego se basa en análisis y razonamientos precisos y es una maravillosa herramienta para desarrollar la capacidad humana en sentido general, al mismo tiempo es en esencia, una vía especial para desarrollar la forma de pensar. En este juego se ha demostrado la no existencia de una estrategia victoriosa para cualquiera de los contendientes, o sea que este juego perfectamente desempeñado es tablas. La demostración es algo compleja, pero a grandes rasgos se basa en que si se conoce tal estrategia cada jugador podrá ponerla en práctica en su turno, por lo que si uno resultara vencedor nos llevaría al caso de que uno ha perdido a pesar de aplicar estrategia ganadora, lo que es un absurdo. Este juego es llamado de "suma cero" o igualado, ya que las oportunidades y la posición de salida es equivalente.

El surgimiento de la Inteligencia Artificial modificó notablemente el paradigma de los juegos de estrategias, en particular el ajedrez. Actualmente son muchas las personas que juegan este maravilloso juego a través de Internet debido a que existen numerosos sitios Web dedicados a este deporte. La comunidad ajedrecística sigue creciendo y regularmente se han realizado obras de caridad como las simultáneas online realizadas por el ex campeón mundial Viswanathan Anand en el 2005 con el objetivo de recaudar fondos para el desarrollo del ajedrez a nivel mundial.

Los desarrolladores buscan satisfacer las expectativas de los usuarios ofreciéndole una experiencia única con cada nuevo juego que salga al mercado. Aunque estos juegos son muy buenos, sus niveles son muy altos para una persona que quiere comenzar a aprender ajedrez Para lograr este objetivo es sumamente importante el desafío ofrecido por la enseñanza del ajedrez.

Situación problémica

En la Universidad de las Ciencias Informáticas actualmente se imparte un curso para la enseñanza del ajedrez como parte de la asignatura de Educación Física. Existe una plataforma de aprendizaje donde se evalúa al estudiante mediante cuestionarios que contienen preguntas e imágenes de tableros de ajedrez con posiciones donde el mismo debe seleccionar de varios movimientos descritos el correcto. Dicha

plataforma no cuenta con una aplicación donde se pueda aprender a jugar ajedrez de forma interactiva por lo que se hace necesario el desarrollo de una aplicación Web que permita resolver la situación anteriormente planteada con el objetivo de que los estudiantes aprendan conceptos básicos del juego de ajedrez.

Problema Científico

¿Cómo desarrollar un juego de ajedrez interactivo para la enseñanza?

Objetivo General

Desarrollar un juego de ajedrez interactivo para la enseñanza mediante la modificación de un programa de ajedrez.

Objeto de Estudio

El proceso de desarrollo de aplicaciones vinculadas al juego de ajedrez.

Campo de Acción

Obtención de una aplicación vinculada a la enseñanza del ajedrez en la Universidad de las Ciencias Informáticas.

Tareas Investigativas

- Revisar fuentes bibliográficas relacionadas con el ajedrez y la Inteligencia Artificial para conocer el estado actual del arte.
- Investigar y estudiar técnicas existentes de la Inteligencia Artificial para aplicarlas en el desarrollo del juego de ajedrez interactivo.
- Estudiar reglamento y estrategias del juego de ajedrez para seleccionar el contenido a utilizar en la confección del juego.
- Consolidar el estudio de los lenguajes seleccionados (PHP, JavaScript, HTML y C) para modificar el motor de ajedrez GNU Chess y confeccionar la aplicación.
- Aplicar las técnicas seleccionadas en el juego de ajedrez para desarrollar la aplicación.

- Estudiar e interpretar el código fuente del motor de ajedrez (GNU Chess) para realizar las modificaciones necesarias que permitan desarrollar habilidades en los estudiantes que incursionan en el mundo del ajedrez.
- Implementar una versión del juego de ajedrez capaz de resolver el problema de la enseñanza del mismo en la UCI.

Idea a Defender

Si se logra modificar el código fuente del motor de ajedrez (GNU Chess) con el propósito de desarrollar habilidades del juego en los estudiantes entonces se obtendrá un software interactivo que contribuya a la enseñanza del ajedrez.

Métodos de Investigación

Para desarrollar esta investigación se utilizaron diferentes métodos científicos, entre los que se destacan el histórico-lógico, analítico-sintético, observación y experimento.

Métodos teóricos:

Histórico – Lógico: Permite estudiar la evolución y desarrollo de algunas técnicas de Inteligencia Artificial para conocer su estado actual.

Analítico – Sintético: Permite concretar y resumir el conocimiento reflejado en los materiales consultados sobre algunas técnicas de Inteligencia Artificial utilizadas en el desarrollo de esta investigación.

Métodos Empíricos:

Observación: Se debe analizar el comportamiento desempeñado por la aplicación de ajedrez controlada por el motor de ajedrez para identificar errores que permitan mejorar su accionar.

Experimento: La realización de pruebas a la aplicación permite verificar y mejorar la calidad de la misma.

El documento consta de un Resumen, Introducción, tres Capítulos, Conclusiones Generales, Recomendaciones, Referencias bibliográficas, Bibliografía consultada, Anexos y Glosario.

En el **Capítulo 1: Fundamentación Teórica** se hace un estudio de todo lo relacionado a las técnicas de inteligencia artificial utilizadas para realizar un juego de ajedrez, su definición, aplicación y programas

existentes a nivel mundial que utilizan estas técnicas. Además se hace referencia a las reglas más importantes del ajedrez reflejadas en una síntesis. En el **Capítulo 2: Motor de ajedrez. Características principales** se exponen las principales tendencias, técnicas y tecnologías utilizadas para el desarrollo de la solución propuesta. En el **Capítulo 3: Propuesta de Solución** se explica cómo se desarrollaron las funcionalidades de la solución propuesta, las pruebas que se le realizaron al software y los resultados obtenidos.

CAPÍTULO 1: FUNDAMENTACIÓN TEÓRICA

El ajedrez es uno de los juegos de mesa más populares del mundo, debido a que se simula una guerra entre los oponentes. No solo se le considera un juego sino un arte, una ciencia y un deporte mental. La enseñanza del ajedrez puede ser útil como forma de desarrollar el intelecto. El ajedrez es jugado tanto de una forma recreativa como competitivamente en clubes, torneos, en Internet, contra computadoras, e incluso por correo, más conocido como ajedrez por correspondencia.

No es un juego de azar, sino un juego racional. El desarrollo del juego es tan complejo que ni aún los mejores jugadores (o los más potentes ordenadores existentes) pueden llegar a considerar todas sus contingencias: aunque solo se juega en un tablero con 64 casillas y 32 piezas al inicio, el número de diferentes partidas que pueden jugarse excede el número de átomos en el universo.

En este capítulo se tratan temas relacionados con la inteligencia artificial y cómo la misma ha sido utilizada a nivel mundial para realizar juegos, específicamente de ajedrez. Además se explican los principales programas para la enseñanza que han sido desarrollados y algunos conceptos, historia y tendencias que son necesarias conocer para el cumplimiento de este trabajo.

1.1 Resumen de las reglas de juego de ajedrez

Cuando el juego comienza, un jugador controla 16 piezas blancas y el otro 16 negras. El color de cada jugador puede elegirse por mutuo acuerdo o al azar. El tablero es colocado de tal forma que ambos jugadores tienen una casilla blanca en sus esquinas derechas respectivas. Las torres, caballos y alfiles más cercanos al rey suelen llamarse *de rey* (ejemplo: torre de rey) y las más alejadas, más próximas a la dama se llaman *de dama* (ejemplo: alfil de dama). Asimismo, el lado donde se encuentran en origen ambos reyes se suele llamar *flanco rey* y el otro *flanco dama*. (1)

Los jugadores se mueven por turnos y siempre solamente una pieza a excepción del enroque, un movimiento especial que se realiza para proteger el rey y sólo se puede hacer cuando este aún no se ha movido ni está siendo amenazado en el momento de ejecutar dicho movimiento. El jugador que juega con las piezas blancas es siempre el que mueve primero. Cada pieza se mueve de forma diferente. (1)

1. Las piezas no pueden saltar, una por encima de la otra (a excepción del caballo, que puede saltar sobre otras, moviendo en "L"), y de la torre, en el enroque.
2. No podemos tener dos piezas del mismo color en una misma casilla.
3. Al capturar una pieza entonces nuestra pieza ocupará esa casilla (excepto en el caso de “comer al paso”).
4. La pieza "comida" se retirará del juego.

A diferencia de las otras piezas, el rey no puede ser capturado. Un jugador puede sin embargo amenazar al rey contrario, lo que se conoce como jaque o bien amenazarlo de tal forma que no haya defensa posible, lo que se llama jaque mate, situación que decide automáticamente la partida, haciendo ganar al que pone en jaque al rey enemigo. Cuando un jugador mueve, su rey tampoco puede colocarse en una casilla en donde se encuentre en jaque.

Es obvio por tanto que el rey es en ajedrez una pieza esencialmente distinta del resto y son las normas respecto a sus movimientos y capturas lo que da al juego su peculiar idiosincrasia y gran parte de su atractivo. Dado que el jaque mate decide la partida, el ajedrez no consiste simplemente, como muchos otros juegos, en capturar todo el material contrario. Por el contrario, muchas piezas de un bando pueden llegar a sacrificarse si de esta forma se puede dar mate al rey contrario.

Un juego de ajedrez no tiene porqué terminar siempre en jaque mate. El juego terminará si ocurre una de estas circunstancias:

- Jaque mate.
- Abandono o rendición.
- Pérdida por tiempo.
- Tablas o empate.

En torneos, y a efecto de computar los resultados, se otorga al ganador de una partida un punto, medio punto a cada jugador que ha hecho tablas, y cero puntos al perdedor de una partida. (1)

1.1.1 Etapas del juego

En el juego del ajedrez se consideran habitualmente tres etapas:

- La apertura, que comprende las primeras jugadas, donde las piezas van saliendo de sus casillas iniciales.
- El medio juego, cuando los dos bandos aún tienen muchas piezas y peones, y estos entran en intenso conflicto.
- El final, donde quedan pocas piezas y peones.

Cada fase de la partida requiere del jugador planteamientos tácticos y estratégicos totalmente distintos. Esto incrementa mucho la complejidad del juego.

1.1.1.1 Apertura

La apertura comprende las primeras jugadas de la partida. Partiendo de la posición inicial, se han clasificado las distintas posibilidades que tienen cada uno de los bandos para conducir sus piezas. Esto se realiza mediante el estudio detallado de cada movimiento lógico de las blancas y, posteriormente, cada posible respuesta (entre las razonables) de las negras, y así sucesivamente.

De forma general, algunas recomendaciones generales para la apertura son las siguientes:

- Adelantar uno o más peones centrales, de ser posible dos casillas, a fin de controlar el centro del tablero.
- Desarrollar las piezas, también controlando el centro.
- No mover demasiados peones, lo que implicaría que se han desarrollado menos piezas.
- Enrocar pronto, a fin de proteger el rey en la esquina del tablero y desarrollar las torres.
- No sacar la dama apresuradamente, ya que podrá ser atacada por los peones y piezas menores adversarias, con la consiguiente pérdida de tiempo al tener que retirarla y guarecerla.

1.1.1.2 Medio Juego

El medio juego empieza tras la apertura y es el momento del juego en el que los dos bandos, que ya han desarrollado sus piezas, entran en un grado máximo de conflicto. Entre jugadores de un nivel ajedrecístico similar, es frecuentemente en el medio juego donde los oponentes luchan por obtener alguna ventaja que

más tarde les asegure ganar la partida. Esta ventaja puede ser de material o de posición. En el caso del material se considera que uno de los dos bandos captura un peón o una pieza, en el caso de posición la ventaja toma muchas formas por ejemplo:

- Control del centro.
- Control de columnas o diagonales abiertas.
- Ventaja de desarrollo.
- Ventaja de espacio.
- Mejor estructura de peones.
- Rey mejor protegido.

1.1.1.3 Final

Por final se entiende la etapa del juego siguiente al medio juego (en caso de que la partida no haya terminado aún) y que se caracteriza por un número escaso de piezas. Esta es otra fase de la partida en la que la técnica es primordial y existen numerosas obras dedicadas a analizar posiciones características en donde el número de piezas y peones es reducido.

1.2 Inteligencia Artificial

Si se pide a varias personas que definan la inteligencia, lo más probable es que nos den definiciones no coincidentes, como por ejemplo:

- Capacidad de aprender a partir de la experiencia.
- El poder de pensar.
- La capacidad de razonar.
- La capacidad de percibir relaciones.
- El poder de comprender.
- Intuición.

El famoso matemático británico Alan Turing, en un documento escrito en 1950, sugirió que la pregunta ¿puede pensar una máquina? era demasiado general y filosófica para ser respondida de forma unívoca. Para concretar, propuso un juego denominado "prueba de Turing". Esta implica a dos personas y un ordenador. Una persona, el interrogador se sienta a solas con la máquina y realiza preguntas en un

terminal, conforme van apareciendo las respuestas, el interrogador ha de averiguar si las respuestas proceden de la máquina o de la otra persona. Si el ordenador tiene un comportamiento inteligente engañará con facilidad al interrogador. Hasta el momento ninguna máquina se ha aproximado a superar esta prueba, a pesar de los más de 40 años de investigaciones en inteligencia artificial, incluso se ha pensado en definiciones menos rigurosas.

1.2.1 Tendencias

La Inteligencia Artificial (IA) trata de conseguir que los ordenadores simulen en cierta manera la inteligencia humana. Se acude a sus técnicas cuando es necesario incorporar en un sistema informático, conocimiento o características propias del ser humano. La definición es una frontera móvil, en la década de 1950 muchos investigadores se esforzaron en ordenadores que pudieran jugar al ajedrez, actualmente baten a jugadores profesionales. En estos momentos hay dos tendencias en la IA, una intenta utilizar los ordenadores para simular los procesos mentales humanos, mientras que la segunda, que es más popular, implica el diseño de máquinas inteligentes, independientemente de cómo piensan las personas. (2)

La primera metodología, simulación, presenta tres problemas: (2)

- La mayoría de las personas tienen problemas para entender y describir cómo hacen las cosas. La inteligencia humana incluye procesos mentales muy difíciles o imposibles de describir y comprender.
- Hay grandes diferencias entre la estructura de un ordenador y un cerebro humano. Por muy potente que sea la computadora no puede tener la capacidad de procesamiento en paralelo de un cerebro.
- La mejor forma de hacer algo con una máquina es muy distinto a como lo haría una persona. Muchos fracasos en IA se deben a intentar copiar a los humanos, en lugar de potenciar las características únicas de los ordenadores.

Con cualquiera de estas dos aproximaciones los científicos se enfrentan a problemas muy difíciles de resolver y de los que ni tan siquiera se sabe si tienen solución.

1.2.2 Inteligencia Artificial en el ajedrez

Uno de los problemas que se intentó resolver fue el del juego del ajedrez, pues tiene unas reglas claramente definidas y unos objetivos inequívocos. Actualmente se conocen varias técnicas de IA para enfrentarse al juego del ajedrez. En general los juegos proporcionan una tarea estructurada en la que es muy fácil medir el éxito o el fracaso. En comparación con otras aplicaciones de Inteligencia Artificial, por ejemplo, comprensión del lenguaje, los juegos no necesitan grandes cantidades de conocimiento. En un primer momento se pensó que se podrían resolver por búsqueda exhaustiva en el árbol del juego, es decir, un árbol que contenga todos los movimientos posibles de ambos jugadores. Considerando por ejemplo el juego de ajedrez, en una partida cada jugador realiza una media de 50 movimientos, con un factor de ramificación medio de 35 posibilidades, por lo tanto para examinar el árbol de juego completamente se tendrían que examinar 35^{100} posibilidades. Resulta evidente que una simple búsqueda directa precisa mucha potencia de cálculo en la práctica, y por lo tanto es necesario algún tipo de procedimiento de búsqueda heurística.

- Búsqueda. Una forma de ganar en un juego es mediante la búsqueda, mirando más allá de las posibilidades generadas por cada movimiento potencial. Los ordenadores son mejores que las personas para este tipo de cálculos y los ordenadores más potentes pueden rastrear todas las posibilidades, en base a la denominada fuerza bruta.
- Heurísticos. Se dice así de los métodos prácticos, se basan en juicios que la experiencia nos dice que son válidos, es como actuamos en la vida diaria. Un programa de ajedrez, por ejemplo podría actuar de la forma: mantenga los peones en la fila del rey tanto tiempo como sea posible.
- Reconocimientos de modelos. Los mejores jugadores humanos de ajedrez recuerdan cientos de modelos críticos en el desarrollo del juego y saben las mejores estrategias aplicables en cada caso. Los programas de juegos también reconocen modelos recurrentes, pero no lo hacen como las personas.
- Aprendizaje de máquinas. Los mejores programas de juegos de mesa aprenden de la experiencia. Si un movimiento es rentable es probable que el programa que aprende lo utilice en futuras jugadas. Si da como resultado una pérdida, el programa lo recordará y evitará su uso en el futuro.

1.3 Las computadoras y el ajedrez

En el siglo XVIII se comenzó a pensar en una computadora que fuera capaz de jugar ajedrez. El español Leonardo Torres y Quevedo construyó, en 1912, un autómata capaz de jugar al ajedrez, llamado *El Ajedrecista*. Cuando surgen las computadoras en la década del 50 del pasado siglo los ingenieros informáticos comienzan a construir programas capaces de jugar ajedrez. 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.

El primer artículo sobre el tema de estrategia contra fuerza bruta fue escrito por Claude Shannon y publicado en 1950 antes de que existiera la primera computadora que jugara ajedrez y predijo acertadamente las dos posibles formas principales de búsqueda de cualquier programa, a las que nombró de “Tipo A”, y de “Tipo B'”. (3)

Los programas “Tipo A”, utilizarían una búsqueda basada en la "fuerza bruta", los cuáles examinarían todas las posibles posiciones de cada rama del árbol de movimientos usando el algoritmo Minimax. Shannon creyó que esto sería muy poco práctico por dos razones: (3)

- Primero, con aproximadamente 30 movimientos posibles en una posición típica de medio juego, Shannon predijo que buscando las 30^6 (más de 700.000.000) posiciones contenidas en los primeros tres movimientos (de ambos bandos, lo que son 6 niveles en el árbol), tardaría aproximadamente 16 minutos, incluso en el caso "muy optimista" que el programa evaluara un millón de posiciones por segundo. Después de esta conjetura, se tardó alrededor de 40 años para conseguir esa velocidad.
- Segundo, se ignoraba el problema de la latencia, ya que el programa trata de evaluar la posición resultante después de todo el intercambio de piezas ocurrido durante todos esos movimientos al final de cada rama del árbol. Los programas de “Tipo A” funcionan así, pero el inconveniente es que se incrementa enormemente el número de posiciones necesarias para el análisis, y de este modo el programa se ralentizaba todavía más.

Shannon sugirió que los programas de “Tipo B” utilizaran una Inteligencia Artificial estratégica para evitar gastar potencia en movimientos malos, y solo analizar las mejores jugadas de cada posición, algo parecido a lo que hacen los jugadores humanos. Esto permitiría al programa analizar las líneas significantes de manera más profunda en un tiempo razonable. (3)

No obstante parece que con la gran capacidad de cálculo existente hoy en día en los ordenadores es preferible la fuerza bruta. La mayoría de los juegos de estrategia, como por ejemplo el ajedrez, usan, hoy por hoy, la fuerza bruta como estrategia de juego, donde analizan todas las posibilidades de juego hasta una determinada profundidad.

A pesar de ello, algunos de estos programas utilizan cortes selectivos los cuáles quedan lejos de ser tan eficaces como lo es la selección que realizan los maestros del ajedrez. Estos son capaces de analizar dos posiciones por segundo estando aptos para competir con programas que pueden analizar en el mismo tiempo millones de posiciones. Así nos acercamos a la visión de lo selectivo que puede llegar a ser el cerebro humano, obteniendo buenos resultados obviamente.

Se puede resumir todo esto con una frase:
“*Los ordenadores juegan a juegos de estrategia generando todas las posibles jugadas por venir, descendiendo a una determinada profundidad y evaluando todas las posibles jugadas, permitiéndoles elegir la que consideren mejor de ellas*”.

1.4 Programas actuales

Programas de libre disposición

Crafty

Crafty es quizás el más popular y conocido programa de ajedrez de código fuente abierto. Su creador es el Dr. Robert Hyatt. El programa de ajedrez "Crafty" fue desarrollado en el año 1983 y actualmente es parte de muchas de las principales herramientas de análisis de partidas de ajedrez (ChessBase, Fritz, HIARCS, entre varios otros). La particularidad de Crafty radica en la aplicación de prácticamente todas las herramientas de desarrollo de software de ajedrez para computadoras en su código fuente. El estudio de

este código ayuda a comprender la aplicación de algoritmos en este tipo de aplicaciones, además de permitir, dada su cualidad de "open source", la posibilidad de probar ideas propias sobre él. (4)

GNU Chess

Programa libre disponible como parte del proyecto GNU de la Free Software Foundation (FSF). El programa está escrito en lenguaje C sobre la base de contribuciones de distintos programadores del mundo y su actual versión es la 5.0. La sección de Frequently Asked Questions (FAQ) del proyecto GNU Chess posee rica información acerca de las contribuciones realizadas al programa. Respecto a su nivel de juego la única referencia concreta que existe es su rating de 2200 puntos jugando en el servidor de ajedrez Free Internet Chess Server (FICS).

Programas Comerciales

Fritz.

Este programa pertenece a la prestigiosa firma alemana de programas computacionales de ajedrez ChessBase. Su sitio Web posee una gran cantidad de información comercial relacionada con este programa, así como publicaciones de eventos y otros productos relacionados con el deporte ciencia, tales como Bases de datos de partidas, catálogos de aperturas, utilitarios para educación.

Este programa se encuentra en su versión comercial 11.0. Su fuerza radica en su nivel de juego, bases de datos incorporadas y módulos de análisis, lo cual lo hace una buena herramienta de entrenamiento para jugadores de todo nivel. Comercialmente ha tenido muchísimo éxito luego de traducirse la versión 4.0 de este software al idioma inglés.

Rybka

Rybka es un programa de ajedrez creado por el Maestro Internacional Vasik Rajlich, actualmente es considerado el programa de ajedrez más fuerte, ya que ha obtenido el Campeonato Mundial de Ajedrez por Computadoras por tres años consecutivos desde el 2007 hasta el 2009.

1.5 Programas para la enseñanza de ajedrez

En esta sección se muestran los principales programas de ajedrez para la enseñanza realizados en el planeta resaltando los tres más importantes, estos son: 50 Chess Games for Beginners, El pequeño Fritz y Total Chess Training.

50 Chess Games for Beginners

“50 Juegos de Ajedrez para Principiantes" es un programa animado de tutoría de ajedrez para los nuevos jugadores que desean mejorar su juego. Proporciona ejemplos plenamente anotados de los juegos de ajedrez que ilustran las buenas tácticas y estrategias, y es particularmente adecuado para los niños o jugadores jóvenes. Se incluye una descripción de la notación en el ajedrez, las aperturas, estándares, clasificación entre otras cosas. (5)

Blind Chess

Es orientado al entrenamiento de ajedrez con los ojos vendados. Funciona con todas las versiones de Windows y en muchos programas para computadoras de bolsillo. (6)

Chess Analysis Pro 7000

El programa es extraordinario en su capacidad de analizar sus propias partidas de ajedrez del modo más útil y detallado disponible. De este modo los jugadores fácilmente pueden mejorar en el juego. (6)

Chess Flashcard Trainer 3.15

Programa para la enseñanza del ajedrez, que automáticamente genera problemas para entrenar y que son descargados gratis de Internet. El programa también anota automáticamente el grado de dificultad de cada ejercicio. (6)

Chessimo

El método de entrenamiento de Chessimo está basado en resolver los ejercicios que el programa produce. Si toma mucho tiempo al resolver los varios ejercicios tácticos, estratégicos y finales, el programa mostrará la solución correcta y le permitirá entrenar de nuevo en una posición similar. (6)

El pequeño Fritz

¡Aprende a jugar al ajedrez de otra manera! En esta aventura original de ajedrez se guía suavemente a los alumnos, sean niños o adultos, por el juego de reyes. ¿Cómo se desplazan las piezas? ¿Qué significa ahogado? ¿A qué se llama oposición? ¿Qué es eso del mate en escalera? De forma clara y expresiva se aclaran esas y muchas dudas más acerca del ajedrez, enmarcadas en la historia del príncipe Fritz, que está sustituyendo a su padre durante las vacaciones. De forma muy fea y desconsiderada, justo en ese momento surge el duelo de ajedrez contra el Rey Negro y el pequeño Fritz no tiene ni idea de cómo se juega. No obstante, con la ayuda del Rey Pintojo, y animado por su valiente prima Bianca y la rata de alcantarilla Leo Listo, el pequeño Fritz acepta el reto y aprende a jugar al ajedrez divirtiéndose, recorriendo diferentes estaciones de juego y entrenándose en el Gimnasio de Neuronas. Al final no solamente conoce las reglas del juego de reyes sino que también se habrá enterado de las pistas y trucos de manera que puede enfrentar al Rey Negro bien preparado, capaz de demostrar lo que ha aprendido. (7) Este programa es dedicado a los niños y ha sido muy reconocido y premiado por su calidad excepcional.

Total Chess Training II

Es una compilación de programas de entrenamiento populares para estudiar todas las fases del juego, la apertura, el medio juego y el final. El paquete de Entrenamiento Total incluye una: Enciclopedia del Medio Juego II, Curso Intermedio de Táctica, Estudios de Mate, Teoría y Práctica de Finales y Entrenamiento en Finales de Ajedrez. Enciclopedia del Medio Juego II es un programa para el estudio de los planes de apertura y técnicas en el medio juego. Una sección teórica que incluye más de 600 partidas/lecturas, cada una de ellas ilustrando típicos planes y métodos en aperturas populares. Una sección especial de entrenamiento con más de 500 ejercicios para que el usuario los resuelva y alrededor de 150 posiciones de entrenamiento en modo prueba. (8) En resumen Total Chess Training incluye más de 11.000 ejemplos de estudio y ejercicios para jugadores de ajedrez con ELO desde 1.700 a 2.400.

1.6 Implementación de programas de ajedrez

Para desarrollar un programa de ajedrez es necesario tener en cuenta los siguientes aspectos:

- Representación del tablero: cómo se representa una posición simple en estructuras de datos.
- Técnicas de búsqueda: cómo identificar los posibles movimientos y seleccionar los más prometedores para examinarlos posteriormente.

- Evaluación de hojas: cómo evaluar el valor de una posición del tablero, sino se hace una búsqueda posterior.

También los programadores deben decidir si utilizan bases de datos de finales, estándares u otras optimizaciones.

1.6.1 Representación del tablero

La estructura de datos utilizada para representar cada posición de ajedrez es clave para el rendimiento de la generación de movimientos y la evaluación de posiciones. Los métodos incluyen el almacenamiento de las piezas en un arreglo y las posiciones de las piezas en una lista.

1.6.2 Técnicas de búsqueda

Los programas de ajedrez consideran los movimientos como un árbol de juego. En teoría, examinan todos los movimientos y entonces todos los contra movimientos a estos y después todos sus respectivos contra movimientos, y así sucesivamente, hasta que llega a una hoja final que es evaluada. Se han ideado varios métodos para aumentar la velocidad de esta búsqueda como por ejemplo:

- Algoritmo Minimax.
- Poda Alfa-Beta.
- Heurística asesina.
- Heurística de movimiento nulo.
- Reducciones de movimiento tardío.

1.6.3 Evaluación de hojas

El último propósito necesario, antes de que se pueda empezar la búsqueda en el árbol, es una función de evaluación. Se realizan algunos movimientos y al terminarlos se debe evaluar cómo de buena es la posición en la que se ha acabado al ejecutarlos.

Las funciones de evaluación típicamente evalúan posiciones en centésimas de peón y consideran el valor del material junto con otros factores que afectan la fuerza de cada bando. Cuando se cuenta el material de ambos bandos, los valores típicos para las piezas son: 1 punto para un peón, 3 puntos para los caballos y

los alfiles, 5 puntos para las torres y 9 puntos para una dama. Por convención, una evaluación positiva favorece a las blancas y una negativa a las negras. (9)

Al rey algunas veces se le otorga un valor arbitrario alto como 200 puntos (artículo de Claude Shannon) o 1.000.000 de puntos (programa de la URSS de 1961) para asegurar que un mate sobrepasa al resto de factores. Las funciones de evaluación tienen en cuenta muchos factores, como la estructura de peones, la pareja de alfiles, la centralización de las piezas, etc. También se suele considerar la protección del rey, así como la fase en la que se encuentra la partida (apertura, medio juego o final).

Se puede conjeturar mucho sobre las funciones de evaluación y los aspectos a tener en cuenta a la hora de definirlas. Los principales objetivos que se tendrán en cuenta, por ser los más razonables, serán la velocidad y el grado de acierto o de eficacia. Cuanto más rápida y eficaz sea la función de evaluación mejor. Obviamente esos dos adjetivos se encuentran enfrentados, una función de evaluación eficaz será más lenta que una “sucia y rápida”. Por ello se ha de diseñar una función de evaluación heurística y no una función exacta que es algo realmente improbable.

Para el ajedrez, existen bases de datos con jugadas finales, en las cuales aparecen posiciones ya marcadas como empates, perdidas o victorias. En casos como estos se deben de aprovechar y no perder tiempo de análisis, puesto que son totalmente eficaces. Ken Thompson, más conocido por ser uno de los creadores del sistema operativo UNIX, fue uno de los pioneros en este tema. Durante mucho tiempo, los programas de ajedrez tuvieron graves problemas al jugar los finales, y eran muy débiles en dicho tramo de la partida, debido a la necesidad de una alta profundidad de búsqueda. Las Bases de Datos de Nalimov es el formato más utilizado en la actualidad y está implantada por la mayoría de los programas de ajedrez como Rybka, Shredder y Fritz. Casi todos los finales de seis o menos piezas, y varios de siete piezas, han sido analizados por completo. Las bases de datos se crean almacenando en la memoria valores de las posiciones, y utilizan estos resultados para podar los finales de los árboles de búsqueda si surgieran de nuevo. El problema principal es que normalmente se trabaja sobre posiciones que no podemos evaluar apropiadamente.

1.6.4 Otras optimizaciones

Otras optimizaciones se pueden utilizar para realizar programas de ajedrez más fuertes. Por ejemplo, las tablas de transposiciones se utilizan para grabar posiciones que ya han sido evaluadas, para evitar recalcularlas. Las tablas de refutación almacenan movimientos claves que "refutan" lo que parece ser un buen movimiento, estas se utilizaron por primera vez en las variantes (ya que un movimiento que refuta una posición es probable que refute otra). Los libros de aperturas ayudan a los programas de computadoras dando las aperturas comunes que son consideradas buenas (y buenos caminos contra las aperturas más pobres).
Con hardware de más velocidad y procesadores adicionales se puede mejorar la capacidad de los programas de ajedrez y, algunos sistemas (como Deep Blue) utilizan hardware especializado en ajedrez en vez de únicamente implementaciones de software.

1.6.5 Estándares

Los programas de ajedrez para computadoras normalmente soportan varios estándares comunes. Casi todos los programas actuales pueden leer y escribir movimientos de ajedrez en el formato PGN y pueden leer y escribir posiciones individuales en formato FEN.

PGN quiere decir Notación Portable de Juego (por sus siglas en ingles "*Portable Game Notation*"), esta permite almacenar partidas de ajedrez, su utilidad es primordial en este deporte ya que consiente tener un registro del juego, como de algún modo en el fútbol se graban los partidos en vídeo, luego se analizan jugada tras jugada para ver el desempeño o saber cómo juega el adversario, además de disfrutarlo y guardarlo para el recuerdo. En el ajedrez es básicamente lo mismo, ¿se ha dado cuenta cuando un jugador hace un movimiento y después hace una anotación?, pues es precisamente eso, al anotarlas se puede analizar el juego y saber porque ganó o perdió la partida, cuál fue el más grave error, dónde no se debió “comer “o cuál fue una magnífica jugada; en fin las utilidades son infinitas. (10)

La Notación PGN se utiliza en todos los programas de ajedrez, lo que lo hace especial ya que debe poder ser interpretado por las máquinas así como por los humanos, en esta notación las jugadas son almacenadas en Notación Algebraica. (10)

En cuanto a la FEN de Forsyth-Edwards (del inglés Forsyth-Edwards Notation), es utilizada para registrar juegos parciales, que empiezan en alguna posición determinada**.** (A diferencia de la Notación PGN que sirve para registrar los movimientos). En los casos de problemas de ajedrez, o soluciones artísticas de estos es esencial y muy útil. (11)
A nivel de desarrollo informático es muy conocida esta notación, es esta la que utilizan los programas en la actualidad, junto con la notación del juego portable PGN, en los sitios de Internet se muestra en caso de que no se puedan utilizar gráficos. Con la notación FEN solo se utiliza un renglón o menos para inmortalizar una posición en el tablero. (11)

1.7 Motor de ajedrez

Un motor de ajedrez es un programa que “sabe” jugar ajedrez y que implementa un lenguaje de comunicación que le permite interactuar con otros programas. Los motores de ajedrez incrementan su fuerza de juego cada año. Esto es en parte por el incremento en la capacidad de procesamiento que permite hacer cálculos más profundos en un tiempo dado. Adicionalmente, las técnicas de programación han mejorado permitiendo a los módulos ser más selectivos en las líneas que analizan y tener una mejor comprensión sobre la posición del tablero.

Muchos programas de ajedrez se dividen en un motor (que calcula el mejor movimiento de la posición actual) y una interfaz de usuario. Muchos motores están separados de la interfaz de usuario y las dos partes se comunican entre sí utilizando un protocolo de comunicación público. El protocolo más popular es el protocolo Xboard/Winboard Communication. Otro protocolo abierto de comunicación de ajedrez alternativo es el Universal Chess Interface. Dividiendo los programas de ajedrez en estas dos piezas, los desarrolladores solo tienen que programar la interfaz de usuario o el motor, sin la necesidad de escribir ambas partes del programa.

1.7.1 Protocolo Xboard/Winboard

XBoard es un programa de ajedrez para los sistemas operativos Unix\Linux desarrollado por Timothy Mann. La versión para Windows es conocida como WinBoard. Xboard/Winboard es un programa Graphic

User Interface (GUI) que permite jugar con motores de ajedrez, jugar por Internet, reproducir partidas PGN, analizar partidas y posiciones y enfrentar motores para saber cual es más fuerte.

Con el protocolo de comunicación Xboard/Winboard se cambia la forma de crear los programas de ajedrez al dividirse en parte gráfica (GUI) y parte inteligente que sabe jugar ajedrez (motor).

1.8 Lenguajes de programación utilizados

Para dar solución al problema que se plantea se utilizan los lenguajes de programación C, PHP, JavaScript y HTML. Por una parte se emplea C en la programación del motor del juego de ajedrez que se utiliza (GNU Chess). Por otra parte la aplicación mediante la cual puede interactuar el usuario con el juego de ajedrez se programa con PHP, JavaScript y HTML por ser una aplicación WEB que se puede utilizar a través de la red.

- C es un lenguaje muy eficiente puesto que es posible utilizar sus características de bajo nivel para realizar implementaciones óptimas. A pesar de su bajo nivel es el lenguaje más portado en existencia, existiendo compiladores para casi todos los sistemas conocidos. Proporciona facilidades para realizar programas modulares y/o utilizar código o bibliotecas existentes.
- PHP es un lenguaje de programación interpretado de propósito general ampliamente usado y que está diseñado especialmente para desarrollo Web y puede ser incrustado dentro de código HTML. Generalmente se ejecuta en un servidor Web, tomando el código en PHP como su entrada y creando páginas Web como salida. Puede ser desplegado en la mayoría de los servidores Web y en casi todos los sistemas operativos y plataformas sin costo alguno. Es un lenguaje multiplataforma. Es libre, por lo que se presenta como una alternativa de fácil acceso para todos. Permite las técnicas de Programación Orientada a Objetos.
- JavaScript es un lenguaje de programación interpretado, es decir, que no requiere compilación, utilizado principalmente en páginas Web, con una sintaxis semejante a la del lenguaje Java y el lenguaje C.

  Es un lenguaje orientado a objetos propiamente dicho, ya que dispone de Herencia, si bien esta se realiza siguiendo el paradigma de programación basada en prototipos, ya que las nuevas clases se generan clonando las clases base (prototipos) y extendiendo su funcionalidad.

Todos los navegadores modernos interpretan el código JavaScript integrado dentro de las páginas Web. Para interactuar con una página Web se provee al lenguaje JavaScript de una implementación del Document Object Model (DOM).

- HTML, (*Lenguaje de Marcas de Hipertexto*), es el lenguaje de marcado predominante para la construcción de páginas Web. Es utilizado para describir la estructura y el contenido en forma de texto, así como para complementar el texto con objetos, tales como imágenes.
  También puede describir, hasta un cierto punto, la apariencia de un documento, y puede incluir un script (por ejemplo JavaScript), el cuál puede afectar el comportamiento de navegadores Web y otros procesadores de HTML.

1.9 Herramientas

Se utilizan herramientas como KDevelop para implementar las modificaciones del motor de ajedrez donde se emplea el lenguaje C y Zend Studio en el desarrollo de la aplicación Web.

KDevelop

KDevelop es un entorno de desarrollo integrado para sistemas Linux y otros sistemas Unix, publicado bajo licencia GPL (General Public License). Su última versión se encuentra actualmente bajo desarrollo y funciona con distintos lenguajes de programación como C, C++, Java, Ada, SQL, Python, Perl y Pascal, así como guiones para el intérprete de comandos Bash. Entre sus características fundamentales podemos citar navegador entre clases de la aplicación, asistentes para generar y actualizar las definiciones de las clases y el framework de la aplicación y completado automático del código en C y C++.

Zend Studio

Zend Studio es un completo entorno integrado de desarrollo para el lenguaje de programación PHP. Está escrito en Java, y está disponible para las plataformas Microsoft Windows, Mac OS X y GNU/Linux. Entre sus características fundamentales podemos citar detección de errores de sintaxis en tiempo real, no requiere la instalación previa de PHP ni del entorno de ejecución de Java, fue diseñado para usarse con el lenguaje PHP; sin embargo ofrece soporte básico para otros lenguajes Web, como HTML, JavaScript y XML.

1.10 Aprendizaje Automático

Una de las características más distintivas de la inteligencia humana es el aprendizaje, esta ha sido una de las principales áreas de investigación de la Inteligencia Artificial desde sus inicios. A través de los años se ha visto un crecimiento acelerado en la capacidad de generación y almacenamiento de información, debido a la creciente automatización de procesos y los avances en las capacidades de almacenamiento de información. Esto ha influido en el desarrollado de gran cantidad de herramientas y técnicas que tienen que ver con el análisis de información.

El aprendizaje humano en general es muy diverso e incluye funciones tales como: adquisición de conocimiento, desarrollo de habilidades a través de instrucción y práctica, organización de conocimiento, descubrimiento de hechos, entre otras. De la misma forma el aprendizaje automático se encarga de estudiar y modelar computacionalmente los procesos de aprendizaje en sus diversas manifestaciones.

El Aprendizaje Automático es una rama de la Inteligencia Artificial cuyo objetivo es desarrollar técnicas que permitan a las computadoras aprender. De forma más concreta, se trata de crear programas capaces de generalizar comportamientos a partir de una información no estructurada suministrada en forma de ejemplos. Es, por lo tanto, un proceso de inducción del conocimiento. (12)

El Aprendizaje Automático tiene una amplia gama de aplicaciones, incluyendo motores de búsqueda, diagnósticos médicos, detección de fraude en el uso de tarjetas de crédito, análisis del mercado de valores, clasificación de secuencias de ADN, reconocimiento del habla y del lenguaje escrito, juegos y robótica.

Algunos sistemas de Aprendizaje Automático intentan eliminar toda necesidad de intuición o conocimiento experto de los procesos de análisis de datos, mientras otros tratan de establecer un marco de colaboración entre el experto y la computadora. De todas formas, la intuición humana no puede ser reemplazada en su totalidad, ya que el diseñador del sistema ha de especificar la forma de representación de los datos y los métodos de manipulación y caracterización de los mismos.

1.10.1 Técnicas más utilizadas

A continuación se presenta las técnicas más empleadas en el Aprendizaje Automático.

- Aprendizaje basado en árboles de decisión***:*** Es uno de los métodos más utilizados para inferencias inductivas. Es utilizado para la aproximación de funciones de valores discretos, robustos frente a datos con ruido y capaces de aprender expresiones disjuntas. Existe una familia de algoritmos de aprendizaje de árboles de decisión que incluye los ampliamente utilizados: ID3, C4.5, y ASSISTANT. Estos métodos buscan un espacio de hipótesis completamente expresivo y de esta manera evitan las dificultades que surgen de espacios de hipótesis restringidos. Su sesgo inductivo es la preferencia de árboles pequeños por sobre los grandes. (13)
- Aprendizaje basado en redes neuronales artificiales***:*** Proveen un método práctico y general para el aprendizaje a partir de ejemplos de funciones reales, discretas e intercalares. Algoritmos como el Backpropagation utilizan el gradiente descendente para ajustar los parámetros de la red de forma que se ajusten mejor a los datos de entrenamiento de entrada-salida. Este aprendizaje es robusto frente a la aparición de errores en los datos de entrenamiento y han sido aplicados con éxito a problemas tales como la interpretación de escenas visuales, reconocimiento del habla y estrategias de control de robots. (13)
- Aprendizaje probabilístico y Bayesiano**:** Provee un acercamiento probabilístico a la inferencia. Está basado en asumir que las cantidades de interés están gobernadas por distribuciones de probabilidad y que las decisiones óptimas pueden ser realizadas por medio de razonamientos sobre estas probabilidades y datos observables. Provee una visión cuantitativa para pesar la evidencia que soportan distintas hipótesis. El razonamiento Bayesiano provee las bases para el aprendizaje de algoritmos que manipulan directamente probabilidades, y un ámbito para analizar cómo operan otros algoritmos que no las manipulan explícitamente. (13)
- Aprendizaje basado en instancias***:*** A diferencia de los métodos de aprendizaje que construyen una descripción general, y explícita de la función objetivo a partir de los datos de entrenamiento, estos métodos simplemente guardan dichos datos. La generalización sobre estos ejemplos se pospone hasta que una nueva instancia deba ser clasificada. Cada vez que una nueva instancia es encontrada, se calcula su relación con los ejemplos previamente guardados con el propósito de asignar un valor de la función objetivo para la nueva instancia. El aprendizaje basado en

instancias incluye el vecino más cercano y métodos de regresión pesados localmente que asumen que las instancias pueden ser representadas como puntos en el espacio Euclideo. Incluyen también los métodos de razonamiento basado en casos, que utilizan una representación más compleja y simbólica de los datos. Los métodos de aprendizaje basados en instancias son denominados “perezosos” pues dilatan el procesamiento hasta que una nueva instancia deba ser clasificada. Una ventaja de este retraso es que no se estima la función objetivo una vez para todo el espacio de instancias, sino que se hace en forma local y diferente para cada nueva instancia a clasificar. (13)

- Aprendizaje lógico inductivo**:** Una de las más expresivas y humanamente legibles representaciones para las hipótesis aprendidas son las reglas Si-Luego-Entonces. Existen varios algoritmos que utilizan este tipo de representación. En particular, uno de ellos que emplea reglas que contienen variables, llamadas cláusulas de Horn de primer orden. Debido a que un conjunto de estas cláusulas puede ser considerado en un programa en el lenguaje lógico de programación Prolog, su aprendizaje es generalmente denominado programación lógica inductiva. (13)
- Aprendizaje por refuerzo: Este tipo de aprendizaje se refiere a como un agente autónomo que actúa en un entorno, puede aprender a elegir acciones óptimas que lo conduzcan a alcanzar objetivos. Este problema genérico cubre tareas tales como el aprendizaje del control de un robot móvil, aprendizaje de cómo optimizar operaciones en una factoría, y aprendizaje de cómo realizar jugadas en juegos de tableros. Cada vez que un agente realiza una acción en su entorno, un entrenador provee un premio o penalización que indica la bondad del estado resultante. Por ejemplo, cuando se entrena a un agente para jugar un juego, el entrenador debe proveer una recompensa positiva cuando el juego es ganado, negativa si se pierde y cero en los otros estados. La tarea del agente es la de aprender a partir de esta recompensa indirecta y retrasada, a elegir secuencias de acciones que produzcan la mayor acumulación de recompensas. Un ejemplo de estos tipos de algoritmos es el Q-learning, que permite adquirir estrategias de control óptimas a partir de recompensas retrasadas, aún cuando el agente no posee un conocimiento inicial del efecto de las acciones en el entorno. Estos algoritmos están relacionados con la programación dinámica, frecuentemente empleada en la resolución de problemas de optimización. (13)

1.10.2 Árbol de decisión

Los Árboles de decisión son una técnica que permite analizar decisiones secuenciales basada en el uso de resultados y probabilidades asociadas. Los árboles de decisión se utilizan en cualquier proceso que implique toma de decisiones, ejemplo de estos son: búsqueda binaria, árboles de juegos y sistemas expertos. (14)

Algunas de las ventajas que proporcionan los árboles de decisión. (14)

- Resumen los ejemplos de partidas, permitiendo la clasificación de nuevos casos siempre y cuando no existan modificaciones sustanciales en las condiciones bajo las cuales se generaron los ejemplos que sirvieron para su construcción.
- Facilitan la interpretación de la decisión adoptada.
- Proporcionan un alto grado de comprensión del conocimiento utilizado en la toma de decisiones.
- Explican el comportamiento respecto a una determinada tarea de decisión.
- Reducen el número de variables independientes.
- Constituyen una magnifica herramienta para el control de la gestión empresarial.

1.10.3 Árbol de juego

Los árboles de juegos son una aplicación de los árboles de decisión donde cada nodo representa una posible elección para uno de los jugadores. Cualquier sucesión de jugadas puede representarse por un camino dentro del árbol de juego. Si el juego acaba siempre después de un número finito de pasos, entonces el árbol tiene un número finito de nodos. Tomando por ejemplo el muy conocido juego del gato donde se considera una función que evalúa una posición del tablero y devuelve un valor numérico (entre más grande es este valor, más buena es esta posición).Un ejemplo de la definición de esta función es considerando el número de filas, columnas y diagonales restantes abiertas para un jugador menos el número de las mismas para su oponente. (14)

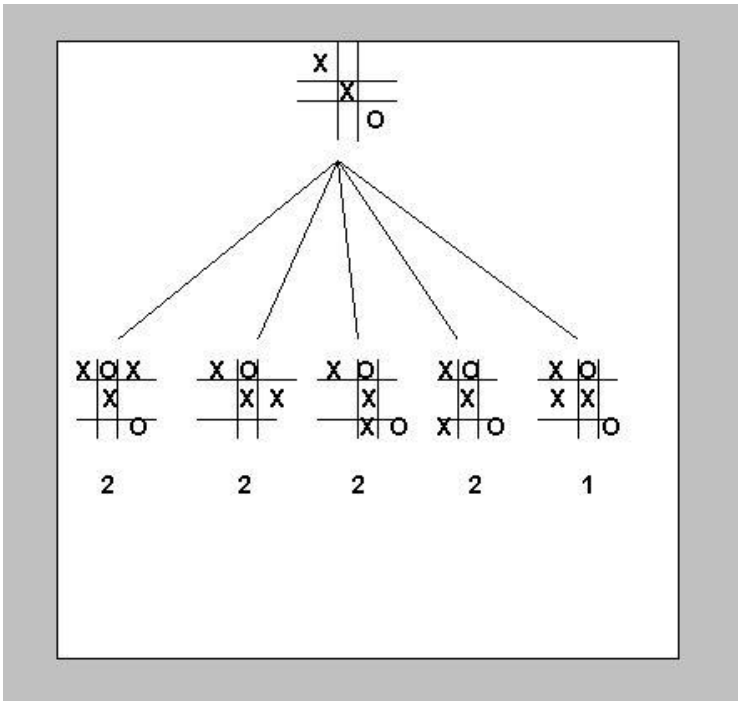

Figura. 1. Ejemplo de un árbol de juego para el juego del gato.

Dada una posición del tablero, para determinar el mejor movimiento siguiente no basta con solo considerar todos los movimientos posibles y las posiciones resultantes. Como se puede apreciar en el ejemplo anterior, las cuatro primeras posibilidades dan todos los mismos valores de evaluación, sin embargo la cuarta posición es sin duda mejor, por lo que se debe mejorar esta función. Aquí se introduce la posibilidad de prever varios movimientos. Entonces la función se mejorará en gran medida, se inicia con cualquier posición y se determinan todos los posibles movimientos en un árbol hasta un determinado nivel de previsión. Este árbol se conoce como árbol de juego cuya profundidad es igual a la profundidad de dicho árbol.

1.10.4 Algoritmo Minimax

Minimax es un método de decisión para minimizar la pérdida máxima esperada en juegos con adversarios, con información perfecta (juegos donde no interviene el azar y en los que cada jugador puede ver toda la información del contrincante, pero desconoce su estrategia) y de suma cero (solamente

se puede ganar, perder o empatar), algunos de estos juegos son las Damas, Ajedrez, Tres en Raya. (15) Expresado en otras palabras el objetivo de la estrategia Minimax es obtener el mejor movimiento para un jugador (Max) donde el otro jugador (Min) intentará minimizar el resultado del juego.

Se designa el turno del jugador 1 como Max, y el turno del jugador 2 como Min, como el árbol empieza con el turno de Max, entonces el árbol estará evaluado de acuerdo a la conveniencia de Max. De acuerdo al árbol del ejemplo que se representa en la figura 1 el mejor primer turno para Max será la cruz en el centro, por lo que el jugador decidirá hacer este movimiento, en esta fase se ve que el turno que sigue es de Min, Min deberá seleccionar la jugada que tenga el menor valor, pues esta será la que perjudique más a Max y convendrá a Min.

Algoritmo Minimax. (16)

1. A partir del tablero *b*, construir un árbol de juego de profundidad *d*, esto es, un árbol con raíz *b*, y que consiste de:
    a) como nodos, tableros.
    b) como aristas de un tablero *b1* a otro tablero *b2*, un movimiento *m* que lleva del tablero *b1* a un tablero *b2*.
2. Para cada hoja, calcular estáticamente su valor.
3. Dar a cada tablero del jugador Max, la mayor evaluación entre los nodos hijos.
4. Dar a cada tablero del jugador Min, la menor evaluación entre los nodos hijos.
5. Elegir el movimiento que lleva del tablero *b* al tablero con máxima evaluación.

Como ejemplo, en la figura 2 el jugador Max en la posición A tomaría la rama que lleva al nodo D, ya que este busca maximizar el valor de la posición y los valores de las opciones son 3, 0 y 8 que corresponden a los nodos B, C y D respectivamente.

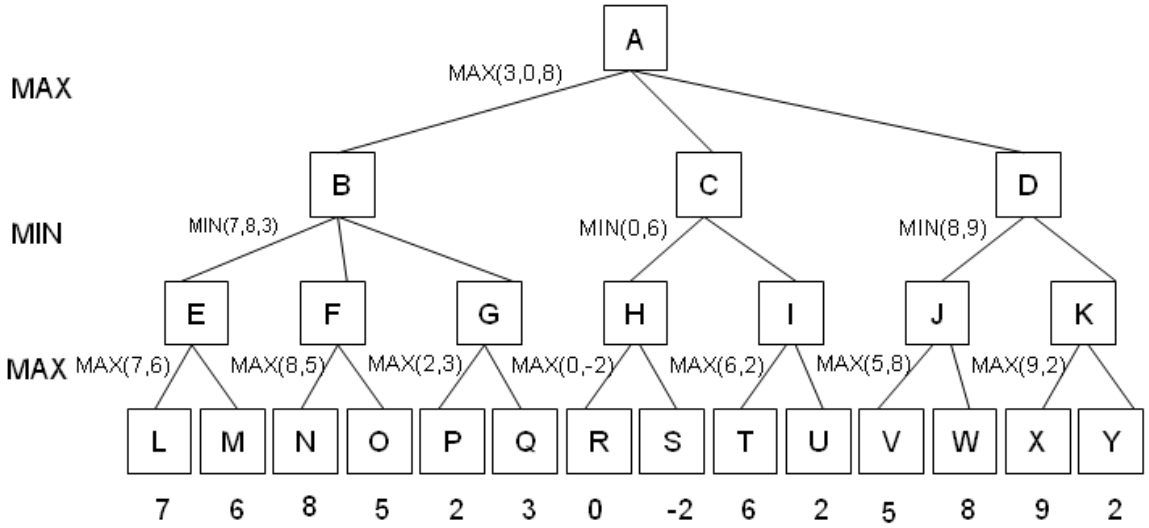

Figura. 2. Ejemplo de un árbol Min-Max.

1.10.5 Algoritmo Poda Alfa-Beta

Haciendo un análisis de la complejidad al Algoritmo Minimax, el número de estados a examinar o evaluar es exponencial en el número de movimientos y el árbol de juego va creciendo de forma considerada, lo que representa un problema. Sin embargo aunque este exponente no se pueda eliminar, existe una posibilidad de reducirlo en gran medida, esto se logra mediante la realización de una poda al árbol Minimax. (17)

La Poda Alfa-Beta es una optimización del algoritmo Minimax. Simplemente se evita examinar (se podan) ramas (soluciones parciales) que se saben son peores que otro umbral o solución conocida. Aplicando la Poda Alfa-Beta, del árbol Minimax se obtienen las mismas jugadas que sin podar, pero recorriendo solo los nodos necesarios. (16)

Algoritmo Alfa-Beta (16)
alfabeta (*b*, α, β)
1. Si el tablero *b* está al límite de profundidad, retornar la evaluación estática de *b*; sino continuar.
2. Sean *b1*,. . ., *bn* los sucesores de *b*, sea *k* = 1, y si *b* es un nodo MAX, ir al paso 3; sino ir al paso 6.

3. Poner α:= max (α, alfabeta (b_k, α, β)).
4. Si α ≥ β retornar β; sino continuar.
5. Si *k* = *n* retornar α; sino poner *k*:= *k* + 1 e ir al paso 3.
6. Poner β:= min. (β, alfabeta (b_k, α, β)).
7. Si α ≥ β retornar α; sino continuar.
8. Si *k* = *n* retornar β; sino poner *k*:= *k* + 1 y volver al paso 6.

La figura 3 muestra la aplicación de la Poda Alfa-Beta al ejemplo de la figura 2. La evaluación del nodo E asegura al oponente (jugador minimizante) una cota superior con valor 7, es decir, el oponente obtendrá 7 o un valor menor en la evaluación de B (dado que los valores están en relación al jugador maximizante, los valores menores a 7 son mejores para el oponente). En este caso, 7 representa el valor beta. A continuación, cuando se examina el nodo N cuyo valor es 8, dado que es mayor que beta = 7, los nodos hermanos de N (en este caso el nodo O) pueden podarse (dado que nos encontramos en un nivel maximizante, el nodo F tendrá un valor igual o superior a 8, y por lo tanto no podrá competir con la cota asegurada beta en el nivel minimizante anterior).

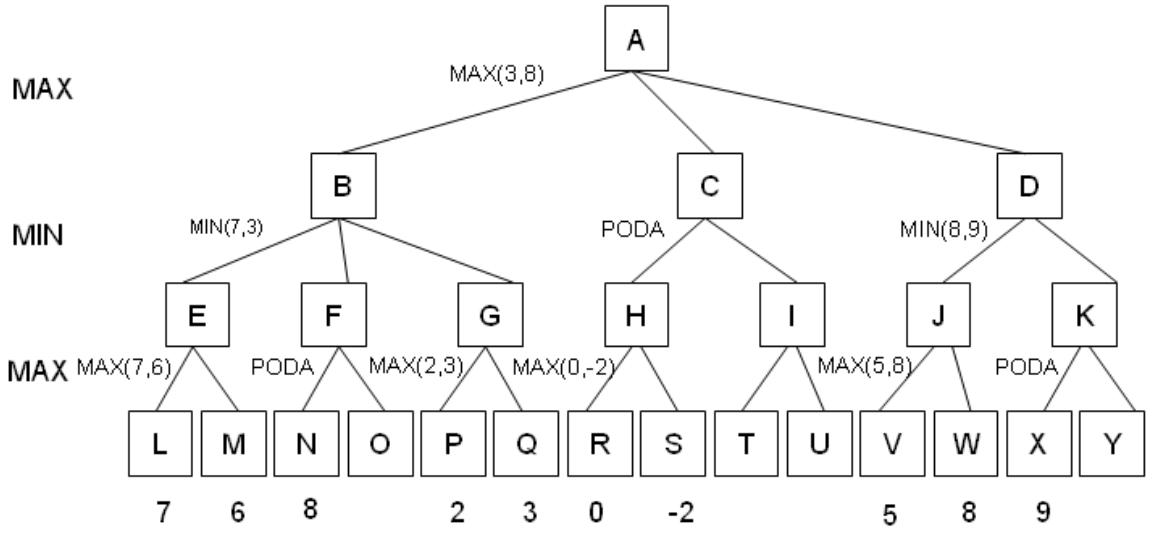

Figura. 3. Ejemplo de un árbol Alfa-Beta.

Luego de evaluar los sucesores de B, se concluye que este nodo asegura al jugador maximizante una cota inferior con valor 3, es decir, obtendrá 3 o un valor mayor en la evaluación de A. En este caso, 3 representa el valor de alfa. A continuación, cuando se examina el nodo H cuyo valor es 0, dado que es menor que alfa = 3, los nodos hermanos de H (en este caso el nodo I) pueden podarse (dado que nos hallamos en un nivel minimizante, el nodo C tendrá un valor menor o igual a 0, y por lo tanto no podrá competir con la cota asegurada alfa en el nivel maximizante anterior).

En un nivel dado, el valor de umbral alfa o beta según corresponda, debe actualizarse cuando se encuentra un umbral mejor. En el ejemplo anterior, al obtenerse el valor 8 del nodo D se puede actualizar el valor de alfa = 3 a alfa = 8. Esto tendría sentido en el caso de que existieran otros nodos hermanos de D que pudieran ser podados utilizando este valor de alfa. Análogamente, al obtenerse un valor de 3 en el nodo G se puede actualizar el valor de beta = 7 a beta = 3. Esto tendría sentido en el caso de que existieran otros nodos hermanos G que pudieran ser podados utilizando este valor de beta.

La efectividad del procedimiento Alfa-Beta depende en gran medida del orden en que se examinen los caminos. Un ordenamiento malo (considerar los peores movimientos primero) causaría que el algoritmo Alfa-Beta se comporte equivalente a un Minimax. Un ordenamiento perfecto (considerar los mejores movimientos primero) causaría que la complejidad del Alfa-Beta sea del mismo orden que un Minimax completo con un factor de ramificación de solo la raíz cuadrada del número de movimientos. Esto significa que se podría explorar el doble de profundidad que con un algoritmo Minimax. (16)

1.10.6 Efecto Horizonte

El Efecto Horizonte se conoce como la incertidumbre de saber, con qué se puede encontrar más allá del “horizonte” (último nivel del árbol de Poda Alfa-Beta), esto se refiere al hecho de que por ejemplo en el nivel horizonte una jugada del tipo captura (en ajedrez) sea buena para un jugador pero sólo en ese momento, porque no se ve lo que puede venir después. En la figura 4 se puede observar un ejemplo. (18)

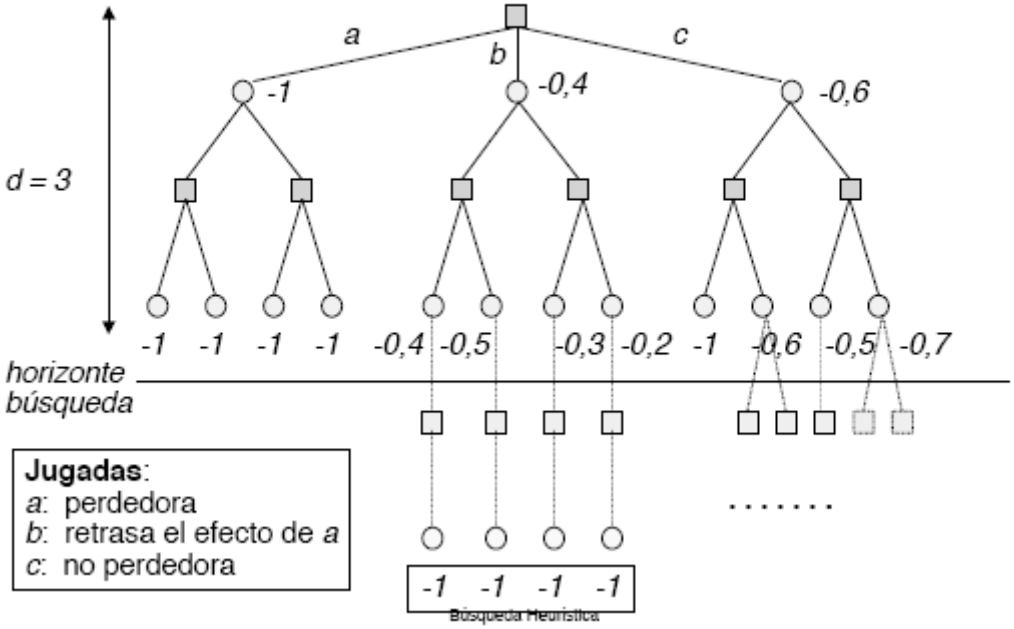

Figura. 4. Ejemplo de árbol de poda alfa-beta.

Considerando [-1,1]; -1: pierde cuadradito, 1: pierde redondo.

En este ejemplo la jugada -1 no se escogería porque se concluye con una jugada perdedora, entonces supuestamente la siguiente mejor jugada seria la *b*, pero si antes se analiza, se puede ver que a una cierta profundidad, se vuelve jugada perdedora, luego la jugada *c* sería la que se debería realizar, pero si no se implementa un método que pueda manejar este caso entonces, el juego podría no ser muy "inteligente".

Una *posición estable* se define como una posición del juego donde ya no existen más jugadas del tipo captura (en caso del ajedrez) o mediante un corte luego de recorrer cierta profundidad, en la búsqueda quiescente. (18)

La búsqueda quiescente es un método de búsqueda que se encarga de hacer una búsqueda de jugadas de solo el tipo captura, de los nodos del nivel horizonte del árbol de Poda Alfa-Beta e irá desarrollándose hasta que el nodo desarrollado sea una posición estable. (18)

Algoritmo búsqueda quiescente (18)

Entrada: jugada //Arreglo de jugadas del último nivel

Salida: número **entero** //que indicará que nodo será tomado en cuenta para la siguiente jugada

JugadasHorizonte_GetJugadasCaptura ()

Si JugadasHorizonte == vacio

    Return setUtilidad (jugada)

Sino

    Mientras <> vacía

        jugada = obtenerjugada (i)

        Si Evaluación (jugada) == jugada captura

            Return Alfa-Beta-Modificado (jugada)

        Fin_Si

    Fin _ mientras

Fin sino

Algoritmo Alfa-Beta-Modificado (18)

Entrada: 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

1.11 Conclusiones del capítulo

En este capítulo se trataron algunos aspectos necesarios para comprender el objeto de estudio que abarca la investigación, como son los conceptos de motor de ajedrez, árbol de juego, las técnicas de búsquedas más generales, inteligencia artificial y aprendizaje automático; cumpliéndose así los objetivos trazados en este primer capítulo.

CAPÍTULO 2: MOTOR DE AJEDREZ. CARACTERÍSTICAS PRINCIPALES.

De todos los motores de ajedrez analizados en el capítulo anterior se decidió escoger el GNU Chess como sistema base para cumplir el objetivo principal trazado, debido a que el mismo posee código abierto, existe una buena documentación facilitando su estudio, y ostenta un código fuente legible y de fácil comprensión para los programadores. Este es uno de los programas más viejos que existen para ordenadores con Unix y los expertos en la enseñanza del ajedrez han recomendado la utilización de motores de ajedrez con potencia de juego no muy fuerte para lograr el aprendizaje en los estudiantes.

En este capítulo se hace referencia al motor de ajedrez GNU Chess en su última versión (5.07), sus características principales así como las técnicas de Inteligencia Artificial que utiliza, los aspectos para la evaluación de la posición del tablero que tiene presente y el proceso de cómo escoger la mejor jugada. Mediante el desarrollo de este capítulo se analiza paso a paso el sistema escogido para ser modificado en beneficio de la enseñanza del ajedrez.

El GNU Chess es un programa de ordenador, creado para jugar ajedrez, y ha sido adaptado a otras plataformas, como Windows. Este programa es una de las partes más viejas del paquete GNU (un sistema operativo compuesto en su totalidad por software gratis), el cual nació en 1984. La primera versión de GNU Chess fue escrita por Stuart Crafcraft. Las versiones consecuentes (hasta la quinta) fueron escritas por John Stanback.

Este juego es gratis, licenciado bajo los términos y acuerdos de la General Public License (GNU)**,** y es mantenido por un equipo de programadores que colaboran entre sí. Es usualmente usado con un programa de GUI (Graphical User Interface), como XBoard.

En 1998-1999 GNU Chess sufrió una metamorfosis en su transición hacia la versión 5 del software. Esta última versión era una reprogramación total del programa, realizada con el propósito de eliminar su confuso código anterior y reemplazar información obsoleta con técnicas de implementación de ajedrez más avanzadas. Entre estas se incluyen bitboards, un algoritmo de búsqueda llamado Principal Variation Search (PVS) (una variación realizada por el profesor Tony Marsland del algoritmo Minimax con Poda Alfa-Beta), entre otras. El autor primario de la versión 5 fue Chua Kong-Sian. Asimismo, utiliza un número

de diversas técnicas para mejorar su desempeño, como un libro de apertura (creado para estudiar juegos maestros), y tablas de transposición, las cuáles se encargan de guardar posiciones previamente analizadas para evitar pérdidas de tiempo re analizando posiciones ya clasificadas.

2.1 Estructura de datos

La principal estructura de datos de GNU Chess es el bitboard (64 bits). Un bitboard, de uso frecuente para los juegos de tableros tales como ajedrez, inspectores y othello, es un tipo de estructura de datos, donde cada pedacito representa una posición o un estado del juego, diseñado para la optimización de la velocidad y/o de la memoria o el uso del disco en cálculos totales. Los pedacitos en el mismo bitboard se relacionan entre sí en las reglas del juego que forman a menudo una posición del juego cuando están tomados juntos. Otros bitboards son de uso general como máscaras para transformar o contestar a preguntas sobre posiciones. Los bitboards ayudan a los programas a analizar posiciones del ajedrez con pocas instrucciones de la CPU y llevar a cabo un número masivo de posiciones en memoria eficientemente. Por ejemplo, 12 bitboards son suficientes para describir el paradero de todas las piezas para ambos bandos, es decir,
BitBoard board.b [2] [6];
Por ejemplo, con un caballo igual a 2 y de color blanco igual a 0 todos caballos se encuentran en la referencia board.b [white] [knight]
#define white 0
#define knight 2
Si se desea verificar que en una casilla se encuentra o no un caballo, seria:
If (BitBoard [B1] & board.b [white] [knight]) {...}
De esta forma se verifica si existe un caballo en la casilla b1 del tablero.

También se utiliza un arreglo para el cálculo de los movimientos simples de un caballo o un rey.
MoveArray [knight or king] [sq]
Esto devuelve un mapa de bits de las casillas a las que un caballo o rey podría moverse de la casilla sq.
Las casillas comienzan en 0 para a1 y enumeradas de izquierda a derecha hasta 63 para h8.

Otro elemento fundamental es la estructura de datos Board.

```
typedef struct
 {
    BitBoard b [2] [7];          /* Piezas por lado (0 - blanco, 1 negro por pieza (1 - peón... 6 - rey */
    BitBoard friend [2];         /* Piezas propias */
    BitBoard blocker;            /* Piezas enemigas */
    BitBoard blockerr90;
    BitBoard blockerr45;
    BitBoard blockerr315;
    short ep;                    /* Ubicación de un paso cuadrado  */
    short flag;                  /* Bandera relativa al enroque del rey */
    short side;                  /* Color del bando a moverse 0 – blanco, 1 - negro */
    short material [2];          /* Valor relativo material total de las piezas por bando sin tener en cuenta el
                                 rey */
    short pmaterial [2];         /* Valor relativo material total de los peones por bando */
    short castled [2];           /* True (1) si el rey esta enrocado */
    short king [2];              /* Posición del rey en el tablero 0 - a1... 63 - h8 */
 } Board;
```

2.2 Generador de movimientos

Este es un método de rotación de bitboard que se utiliza generalmente en los programas que usan bitboards. Estos bitboards giran las posiciones a 90 grados, 45 grados, y/o 315 grados. Un bitboard típico tendrá un octeto por fila del tablero del ajedrez. Con estos bitboards es fácil determinar ataques de la torre a través de una fila, usar una tabla puesta en un índice por la casilla y las posiciones ocupadas en la fila (porque los ataques de la torre paran en la primera casilla ocupada). Girando el bitboard 90 grados, los ataques de la torre a través de una columna se pueden examinar de la misma manera. Girando el bitboard 45 grados y 315 grados se producen los bitboards en los cuales las diagonales son fáciles de examinar. La dama puede ser examinada combinando ataques de la torre y el alfil. Los bitboards girados se deben ver como una optimización avanzada.

2.3 Técnicas utilizadas

Las técnicas que a continuación se explican son utilizadas por el motor de juego GNU Chess para realizar optimizaciones que facilitan la búsqueda en el árbol de juego.

2.3.1 Heurística del Movimiento Nulo

Una de las técnicas presente en el motor de juego es el Movimiento Nulo, o *Null-move forward pruning*. Esta técnica permite reducir de forma drástica el factor de ramificación con un cierto riesgo de perder información importante. La idea es dar al oponente una jugada de ventaja, y si su posición sigue siendo buena, (alfa mayor que beta), se asumirá que el alfa real seguirá siendo mayor que beta, y por tanto se poda esa rama y se continúa examinando otros nodos. Aun así, esta técnica, a parte del riesgo de perder información, tiene otros dos inconvenientes: inestabilidad en la búsqueda (pues los valores de beta pueden cambiar), y que no se suelen permitir dos movimientos nulos seguidos.

2.3.2 Ordenamiento estático de movimientos

Esta forma de ordenar movimientos es una heurística dependiente del tipo de juego. Desde un punto de vista teórico, las heurísticas independientes son mucho más satisfactorias, pero para un programa determinado puede que se necesite heurísticas más realistas que proporcionen un resultado mejor.
En el motor de ajedrez GNU Chess el ordenamiento estático se realiza de forma tal que para ordenar los movimientos se consideran primero las capturas y las promociones, luego las jugadas de mates y finalmente los movimientos al centro del tablero.

2.3.3 Libro de apertura

El libro de apertura se implementa como un simple archivo binario que consta de un conjunto de registros secuenciales de hashtype. Esta estructura de datos está dividida en dos partes, una de 64-bit Hashtype y una de 32-bit para la puntuación.

El libro binario almacenado en book.dat se compila a partir del archivo book.pgn usando el comando "add book book.pgn" dentro de un conjunto secuencial de registros binarios en el formato que se ha descrito anteriormente. book.pgn es simplemente un conjunto de registros de juegos en formato de notación de juego portátil (PGN). Un conjunto de juegos maestros se pueden usar o aperturas programadas específicas para que de este modo se convierta en un libro de apertura usuario-configurable.

2.3.4 Tablas de transposición (hash)

La tabla de transposición (hash) del GNU Chess es simplemente un espacio de memoria donde la información sobre posiciones se almacena. En esta versión del programa hay dos tipos de tablas hash: general y para el peón. El tamaño de la tabla hash general es controlado por la variable HASHSLOTS. Asimismo, el tamaño de la tabla hash peón es controlado por la variable PAWNSLOTS. El tamaño de HashTable puede ser controlado desde la línea comando o a través del comando interactivo “hashsize”. Normalmente las búsquedas en el medio juego se aceleran en un 25% -50% por la tabla hash general y por mucho más en finales de juego donde existen pocas piezas (debido a que muchas de las posiciones dejan de ser almacenadas en la tabla hash.)

La evaluación de peón es tradicionalmente costosa, porque se tiene en cuenta muchos criterios para evaluar. La tabla hash de peón memoriza todas las estructuras de peón en la búsqueda. Normalmente la evaluación de la estructura de peón (sin referencia a las piezas) se puede calcular mediante una tabla de búsqueda en un 90-99% del tiempo. Por otra parte, la estructura de peón se refiere a las piezas que deben ser re calculadas para cada posición.

Como ejemplo de la utilidad de la tabla hash se puede imaginar un juego donde las blancas muevan A, las negras respondan con B y las blancas continúan con C. Si el juego ha ido al revés (C, B, A) tendremos la misma posición de nuevo y no es necesario volver a buscarla otra vez si se tiene almacenado lo que se calculó como resultado de la posición en A, B, C.

Por tanto al final de cada llamada a la función Alfa-Beta se debe almacenar el resultado y al principio de las mismas se debe de realizar una búsqueda en la tabla hash para verificar si realmente se debe de calcular algo o no. En la tabla hash por tanto se deben de almacenar: la posición, el valor de dicha posición, la profundidad con la que se llegó a ese valor y el tipo de valor. El tipo de valor indicará si el valor almacenado es un valor Minimax exacto o si por el contrario es una aproximación por encima o por debajo del valor verdadero. Si el valor devuelto por Alfa-Beta no está entre alfa y beta, no es un valor exacto. Si el valor es mayor que beta es una aproximación por debajo del valor real y si el valor es menor que alfa es una aproximación por encima del valor real. Esto es importante pues se pueden llegar a posiciones iguales con unos valores de alfa y beta diferentes.

Lo primero que Alfa-Beta hace es verificar en la tabla si la posición ya ha sido calculada alguna vez. Si se devuelve un valor distinto de NULL, se tiene información disponible acerca de la posición. No obstante se debe de verificar que la información resultante sea útil, pues por ejemplo la información obtenida puede venir dada de una operación anterior que se realizó con una profundidad distinta, en cuyo caso no será útil. De otro modo si es un valor exacto se puede devolver inmediatamente, pues sabemos que el valor real de la posición es inferior o igual al almacenado. Si el valor resulta menor o igual que alfa, también puede ser devuelto, porque tan solo interesan los valores entre alfa y beta.

Si el valor es mayor que alfa pero inferior a beta, se puede ajustar beta para ahorrar tiempo en la siguiente llamada a Alfa-Beta. Para que la tabla hash sea lo más eficiente posible, se necesita que sea capaz de almacenar cuantas más entradas posibles mejor, con un tiempo de almacenamiento y búsqueda pequeños que es proporcionado por la propia función de hashing.

2.3.5 Mejoras en la búsqueda

Todas las técnicas que se han mencionado y explicado intentan reducir el número de nodos a analizar mejorando el orden de los movimientos. Existen otras clases de mejoras que persiguen el mismo objetivo pero de una forma distinta. Estas mejoras intentan explotar la naturaleza del algoritmo Alfa-Beta, que tiene que buscar menos nodos cuando la ventana Alfa-Beta sea menor.

2.3.5.1 Windowing

Windowing es la más simple de las técnicas que se presentan a continuación. Cuando se trabaja a profundidad iterativa, siempre se sabe cuál es el valor de la iteración anterior. Así pues se intentará reducir el esfuerzo en la búsqueda utilizando una ventana Alfa-Beta menor. En lugar de utilizar -INFINITY y +INFINITY se puede utilizar un intervalo ultimovalor- WINDOW, ultimovalor+WINDOW. Si el valor real del Minimax en esta iteración se encuentra en esta ventana tendremos que buscar menos nodos que con una ventana más amplia. En el caso de que no sea así tendremos que aumentar la ventana para realizar la búsqueda con una más grande.

2.3.5.2 Principal Variation Search (PVS)

Principal Variation Search (PVS) es una modificación del algoritmo Minimax con Poda Alfa-Beta efectuada por el Profesor Tony Marsland. Windowing suele ser la mejor opción, no obstante se ve restringida por el nodo origen. PVS intenta profundizar realizando predicciones en la ventana Alfa-Beta en cada nodo. La idea básica radica en que el valor localalpha se encontrará en su valor máximo después de los primeros movimientos, PVS intentará explotar esto llamando a la función Alfa-Beta recursivamente con diferentes parámetros que el estándar Alfa-Beta. En PVS se asigna a alfa el valor localalpha y a beta el valor localalpha+1, quedando algo como valor = -alphabeta (p, d-1,-(localalpha+1),- localalpha); se espera que en esta llamada se falle por debajo del valor alfa pues se parte de la creencia que se ha encontrado el mejor movimiento. Si no se produce este fallo debemos volver a llamar a Alfa-Beta con los valores normales. A PVS también se le conoce como la técnica NegaScout.

2.3.5.3 Búsqueda Quiescence

Algunas veces no resulta adecuado evaluar una posición si es demasiado caótica. El significado que se le puede dar a caótica depende del tipo de juego. En ajedrez, por ejemplo, si un movimiento implica perder una torre; pero puede provocar que después se pueda convertir un peón en dama, que desemboque en ganar la partida. Si se llama a la función de evaluación que se tiene quizás concluiría que es una jugada negativa por perder la torre, mientras que se sabe que en realidad no lo es. Por ello hay que preparar la

función de evaluación para que una vez que se le llame, para determinados movimientos (muy pocos necesariamente para no perder rendimiento), realice una búsqueda mucho más minuciosa y detallada de lo normal.

2.4 Evaluación de posiciones

Para llevar a cabo la evaluación de una posición determinada en el tablero, GNU Chess contempla varios aspectos de la evaluación en términos ajedrecísticos y los lleva a la práctica mediante la programación. Es muy importante comprender la magnitud de la sección que a continuación se explica porque constituye un apoyo invaluable para emitir una sugerencia al usuario.

GNU Chess posee un evaluador que consta de una serie de rutinas de puntuación como ScoreP (), ScoreN (), ScoreB (), que corresponde a la puntuación de cada tipo de pieza. La rutina evalúa todas las piezas de ese tipo (P - peón, N - caballo, B - alfil, etc.) sobre el tablero actual y devuelve una puntuación.

Normalmente se utiliza un bucle de la forma

```
short sq;        /* Posición de la pieza de ese tipo*/
short s;         /* Puntuación que se devuelve del tipo de pieza analizada*/
BitBoard b;      /* Almacena el bitboard que representa las posiciones del tipo de  pieza */
s = 0;
b = board.b[side] [knight];
   while (b) {
     sq = leadz (b);
     CLEARBIT (b, sq);
     If (pieza en sq está sujeta a algún criterio de evaluación)
          s += UNA_BONIFICACION_O_PENALIDAD;
   }
   Return(s);
```

Esta rutina sitúa cada pieza en el mapa de 64 bits donde cada casilla correspondiente a las 64 se modifica a 1 si hay una pieza. Por ejemplo, en la posición de apertura, board.b [white] [bishop] tendría el 3ro y 6to

bit del orden bajo (del mapa de bits) correspondiente a la localización original de los alfiles en un juego en C1 y F1. Asimismo los valores BitPosArray [C1] y BitPosArray [F1] se puede utilizar para devolver cantidades de 64-bits para el control especifico de validaciones como:

```
If (BitPosArray [A1] & board.b [side] [bishop])
        s += UNA__PENALIDAD_POR_ALFIL_EN_ESQUINA_A1
```

El código escrito de evaluación viene muy natural utilizando estos métodos. Intentar evitar demasiados controles específicos de casillas es costoso. Ideas como se muestra en la rutina CTL () se puede utilizar para comprobar si la colocación de una pieza sobre las casillas son favorable o desfavorable.

Las evaluaciones primarias se realizan con la rutina Evaluate (). Algunos detalles calculados se consideran muy importantes, como la evaluación del rey y del peón. Entonces, una "evaluación perezosa" es verificada para lo que se puede ahorrar tiempo. De lo contrario otras piezas también son evaluadas.

2.4.1 Rasgos de evaluación básicos

Material

Material es la suma de piezas para cada lado. El valor material es por lo general la parte más influyente de la evaluación. Por lo general esto es la suma de un valor constante para cada pieza presente. Los peones son considerados la unidad baja con un valor de 100).

Otras consideraciones materiales

En el ajedrez, se sabe que ciertos rasgos materiales proporcionan una ventaja, como el par de alfiles. El GNU Chess aumenta los valores de las torres cuando hay menos peones sobre el tablero y a los caballos cuando hay muchos peones.

Otros factores que afectan la evaluación material son:

- Puntos adicionales para el par de alfiles (los alfiles se complementan el uno al otro, controlando los cuadros de diferente color).
- La disminución de los peones torres y aumentando el valor de los peones centrales.

- Penalidad por no tener ningún peón, ya que esto hace más difícil de ganar al final.
- Se penalizan los desequilibrios materiales.

Material insuficiente

GNU Chess utiliza código para descubrir combinaciones de tablas o probablemente balance de tabla. Por ejemplo, rey y alfil contra rey es tabla, tal cual rey y caballo contra rey, y rey y dos caballos contra rey entre otras. El Equilibrio Material es el término de evaluación que predomina.

Movilidad

La movilidad es una medida del número de opciones legales que tiene un jugador en una posición dada. A menudo se usa como un término en la función de evaluación de los programas de ajedrez. Se basa en la idea de que, el que más opciones tiene a su disposición, más fuerte será su posición. En un estudio de 380 torneos de ajedrez en el que el balance material sigue siendo igual pasada la jugada, se puso de manifiesto una clara correlación entre la movilidad de un jugador y el número de juegos ganados.

Cálculo de la movilidad.

En el GNU Chess el cálculo de la movilidad se efectúa pieza por pieza, y la bonificación por movilidad posible siempre es la misma para cada tipo de pieza. Además, si una pieza puede moverse a la casilla de otra pieza de su mismo bando, también se cuenta, aunque no sería un movimiento legal, es la protección de una pieza y, por tanto, todavía tiene un papel útil.

Piezas atrapadas

Las piezas atrapadas son ejemplos de una movilidad pobre. Son colocadas en una mala casilla y, a menudo, son extinguidas. Un ejemplo típico de una pieza atrapada es un alfil blanco en h7 bloqueado por peones negros en f7 y g6. La mayoría de las veces, es finalmente capturada.

Otro de los ejemplos incluye:

- Un caballo blanco sobre H8 en presencia de un peón negro en h7 o bien en f7.
- Un alfil de color blanco en h6 con peones negros en g5 y f6.
- Un caballo blanco sobre h7 con peones negros en el g7 y h6.

• Una torre blanca en h1/g1/h2/g2 con un rey blanco en f1 o g1.

Seguridad del rey

Una buena evaluación de la seguridad del rey es una de las tareas más arduas en la escritura de una función de evaluación, pero también la más gratificante. Los temas se colocan en este orden aproximado de (aplicación) dificultad.

Peón escudo

Cuando el rey se ha enrocado, es importante preservar los peones junto a él, con el fin de protegerlo contra el asalto. En términos generales, es mejor mantener los peones inmóviles o, moverlos una casilla. La falta de un peón blindaje merece una sanción, más aún si hay una columna abierta junto al rey.

Peón tormenta

Si el enemigo está cerca de los peones del rey, puede haber una amenaza de abrir una columna, incluso si el peón escudo está intacto.

Control de casillas

Es importante recopilar información sobre el control de las casillas cerca del rey enemigo. Hay ciertas posiciones típicas que tienden a requerir conocimientos adicionales. Por ejemplo, el fianchetto (una vez un alfil esta en fianchetto, no se debe intercambiar, sobre todo cuando se trata de una parte de la formación defensiva del rey).

Estructura de peones

Estructura de peones es un término utilizado para describir las posiciones de todos los peones en el tablero, haciendo caso omiso de todas las demás piezas. La estructura de peones abarca una amplia gama de ideas, de la forma general de los peones (como abierto o cerrado) a las características específicas de cada uno.

La evaluación completa de la estructura peón puede ser bastante caro. Por ese motivo, muchos programas incluyendo GNU Chess utilizan una tabla hash adicional para peón solamente.

Torre en 7ma u 8va fila

Se trata de un término común de evaluación. En el ajedrez, con una torre en séptima fila es una posición muy fuerte, ya que puede atacar muchos peones contrarios y tenderle una trampa al rey. Especialmente la situación está dominada con el control absoluto de la séptima fila con dos grandes piezas ayudando y con el rey atrapado en la 8va fila, esto puede tener un efecto devastador.

Otros programas solamente dan una bonificación de torre en la séptima fila, si hay peones oponentes en el 7ma, o el rey oponente está en 8va.

Torres en columnas abiertas o semi-abiertas

Una columna abierta se define generalmente como una columna sin peones en la misma; semi-abierta es una columna que contiene sólo peones enemigos. En ambos casos, una torre que ocupa esa columna tiene una mayor movilidad vertical, así como una oportunidad para penetrar en el terreno enemigo, situándose en la 7ma fila.

Alfil malo

Es un alfil cuya movilidad está restringida por sus propios peones. A diferencia de los problemas de movilidad normal, esta situación es semi-permanente (más si la posición está bloqueada), lo que justifica el hecho de singularizar la evaluación como otro término.

Evaluación perezosa

La evaluación perezosa consiste en dividir todas las tareas realizadas por la función de evaluación en (normalmente dos) etapas. Si después de realizar todas las tareas para una determinada fase beta, por puntuación supera un cierto margen, o si cae por debajo de alfa por el mismo margen, la puntuación es devuelta y no se lleva a cabo la evaluación.

La evaluación perezosa puede ser considerada como un paso más del principio Alfa-Beta. Si algunas partes de la función de evaluación se omiten debido a esta heurística, esto significa que un bando ya está ganando "la evaluación del juego”.

2.5 Obtención de la mejor jugada

El proceso de seleccionar la mejor jugada se realiza explorando los nodos del árbol de juego. El GNU Chess, como todos los motores de ajedrez, tiene un tiempo establecido para dar respuesta a la jugada de su oponente, mientras no finalice este tiempo, el programa realiza una búsqueda en profundidad en el árbol, mediante el algoritmo Principal Variation Search y la búsqueda Quiescence o NegaScout, todas analizadas en este capítulo. La posición actual en el tablero es la raíz original del árbol, posteriormente se va construyendo el árbol utilizando el generador de movimientos de bitboards que concibe todas las jugadas validas posibles, estas forman parte del siguiente nivel del árbol de juego. A continuación se procede a ordenar dichos movimientos con la ayuda de la función de evaluación o heurística, dónde se aplican las tácticas, estrategias del juego, para alcanzar el movimiento más óptimo y que sea reflejado en la evaluación de la posición de ambos bandos en el tablero. Esto permite realizar una poda eficiente al árbol de juego descartando las jugadas más malas. El proceso de generar los movimientos válidos, ordenar dichos movimientos y seguir la búsqueda de la mejor jugada se repite hasta cumplir el tiempo permitido para realizar la búsqueda, u obtener la mejor jugada sin finalizar este tiempo.

2.6 Principales comandos de GNU Chess

Un comando (calco del inglés *command*, «orden, instrucción») es una instrucción o mandato que el usuario proporciona a un sistema informático, desde la línea de comandos (como una *shell*) o durante la ejecución de un programa. Este último caso es aplicable al motor de ajedrez GNU Chess, el cual con la ayuda de estos comandos puede interactuar con la interfaz de usuario (GUI) sin necesidad de preocuparse por el diseño de la misma, de ahí la importancia de estas instrucciones. A continuación una explicación detallada de cada uno de estos comandos por la relevancia que tienen en la aplicación final:

Quit → para terminar el programa.

Exit → en el modo de análisis del motor se detiene, en otro caso finaliza el programa.

Book → para utilizar el libro de aperturas, dentro de este comando se encuentran implícitos los cuatro siguientes:

- On → para activar el libro.
- Off→ para desactivar el libro.
- Best → mejor jugada del libro.
- Random → cualquier jugada aleatoria del libro.

Pgnsave → salvar el juego en formato PGN.

Pgnload → cargar partida en formato PGN.

White→ le otorga el turno a las blancas.

Black→ le otorga el turno a las negras.

Go→ el motor efectúa el movimiento del bando en turno.

New→ para comenzar un nuevo juego.

Hash → permite el funcionamiento de la tabla de hash.

On→ activa la tabla hash para mejorar la velocidad de búsqueda.

Off→ deshabilita la tabla hash.

Null→ para utilizar la heurística del movimiento nulo en la búsqueda aquí también se recurre a los comandos On y Off para activar y desactivar respectivamente.

Xboard→ para hacer uso del protocolo XBoard/WinBoard.

Depth → define la profundidad con la cual se va a realizar la búsqueda de la jugada.

Undo→ permite deshacer un movimiento.

Setboard→ permite modificar el tablero a una posición especificada por la FEN que se introduce.

2.7 Conclusiones del capítulo

En este capítulo se ha tratado sobre los aspectos fundamentales del motor de juego GNU Chess. Se explica cómo es su estructura de datos, su sistema de evaluación para una posición determinada, así como los comandos principales con los cuales funciona y las técnicas de inteligencia artificial que utiliza para realizar y optimizar la búsqueda en el árbol de juego. Todos estos aspectos contribuyen a una mejor compresión del sistema con el cual se trabaja.

CAPÍTULO 3: PROPUESTA DE SOLUCION

Es imposible pensar en un trabajo investigativo serio sin que se realicen pruebas de software y se valoren los resultados obtenidos en la solución propuesta. La interfaz que interactúa con el usuario y le permite una mayor interacción con él a la hora de jugar ajedrez, le brinda al estudiante una serie de sugerencias en determinados parámetros de la evaluación de la posición explicados en el capitulo anterior. Además de contar con un diseño de buen gusto, esta aplicación se caracteriza por tener una usabilidad apropiada y es una herramienta mediante la cual se puede practicar la enseñanza del ajedrez a distancia. No se puede dejar de mencionar su soportabilidad. De esta forma se reflejan los aspectos mas importantes de la aplicación Web, pero no se puede menospreciar al motor de ajedrez, vinculándose con la interfaz, que le permite realizar todas sus funcionalidades. En más de una ocasión pensadores y filósofos han expresado a lo largo de la historia que lo que realmente interesa no es la apariencia sino lo que se lleva por dentro. En el presente trabajo se cumple lo que dijeron estos pensadores porque la modificación del motor de ajedrez GNU Chess cuyo código fuente es de libre acceso es el mayor aporte en la solución. La interacción entre el motor y la interfaz y la propia interfaz son otros aspectos significativos de la aplicación final. En este capítulo se realizará un análisis profundo y detallado sobre todos estos aspectos.

3.1 Modificación en el motor de ajedrez GNU Chess

Las sugerencias brindadas al estudiante corresponden a una evaluación de la posición actual del tablero donde se analizan las piezas del bando en turno y se ilustra una valoración cuantitativa. Los aspectos más importantes a tener en cuenta para ofrecerle consejos y advertencias al bando en turno son: ventaja material, seguridad del rey, estructura de peones, desarrollo de piezas, control del centro.

Primeramente se implementa una función llamada EvaluaConMensajes (int side, int alpha, int beta) donde recibe como entrada el bando (blancas o negras) al cual se le desea emitir las sugerencias, y los límites de evaluación alpha y beta respectivamente. En esta función se comprueba si la posición actual del tablero puede ser especialmente tratada, por ejemplo, si los límites indican que se está analizando un posible jaque mate entonces solamente se devuelve la ventaja material debido a que si dicho valor es infinito(suficientemente grande) quiere decir que se está en presencia de un jaque mate. En caso de estar en una posición final de rey y peones contra rey se ejecutaría la rutina RPR (). Si un bando tiene el rey

solo y el otro posee piezas que no son peones se llama a la función ReySolo (). También se verifica que la posición no sea tablas por causa de material insuficiente. Y en otros casos se ejecutan funciones que analizan la ventaja posicional de las distintas piezas del bando que se tiene en cuenta.

3.1.1. Estructura de peones

Los peones quizás no sean piezas, pero su evaluación es muy compleja debido a la gran de variedad de situaciones que pueden presentarse y sin ellos se hace prácticamente imposible ganar la partida, es por ello que la estructura de peones es esencial. La función implementada para este caso recibe el bando a evaluar, se chequean todos los parámetros de evaluación ya tratados con anterioridad y se imprimen sugerencias que alerten al jugador sobre una situación determinada, brindando como dato adicional la casilla donde se encuentra el peón. Además se tiene en cuenta la fase en que se encuentre el juego (apertura, medio juego, final) pues de aquí depende el peso que puedan tener en la posición. La tabla hash o de transposición especial de peones ayuda a recordar las posiciones previamente analizadas para disminuir el tiempo de ejecución y no tener que realizar una búsqueda en el árbol de decisión de la jugada correcta o mejor dicho en el árbol de juego donde este último es una aplicación particular del primero.

Las sugerencias que se imprimen sobre la estructura de peones son las siguientes:

```
Printf ("Peón %s...\n", casillas [sq]);
        Printf ("Peón pasado\n");
Printf ("Peones doblados en la columna %c\n", columnas[i]);
Printf ("Peón aislado en la columna %c\n", columnas[i]);
Printf ("El Peón f6 junto a la reina es muy fuerte contra el rey enrocado\n");
Printf ("Los peones pasados conectados son fuertes en la fila 6 y 7\n");
Printf ("Los peones centrales d2 y e2 son importantes en el control del centro, estos están bloqueados\n");
Printf ("El Rey enemigo está fuera del alcance del peón pasado %s\n", casillas [sq]);
Printf ("Cuando ambos reyes están enrocados es importante desarrollar la tormenta de peones\n");
```

Cada una de estas sugerencias se analizan desde el punto de vista de los dos bandos, por ejemplo, cuando se habla de los peones centrales d2 y e2 también se tienen en cuenta sus homólogos de las negras d7 y e7.

3.1.2 Seguridad del Rey

La pieza más importante y compleja en un juego de ajedrez es el rey y es precisamente porque en todo juego de ajedrez se debe suministrarle protección. Los aspectos fundamentales a tener en cuenta para la seguridad del rey son, el enroque, la estructura de peones para proteger al rey y si la misma está bajo ataque. Por lo tanto se implementa una función para obtener sugerencias respecto a la seguridad del rey, las cuales son:

printf("*El rey está enrocado\n*");

Este mensaje se imprime en caso de que el rey ya haya sido enrocado y en caso contrario se imprime el mensaje "*El rey no se ha enrocado*".

Printf ("*El rey no se puede enrocar, generalmente el rey se pone en seguridad mediante el enroque desarrollando además la torre del enroque\n*");

Printf ("*No hay peones propios en la columna del rey\n*");

Printf ("*No hay peones enemigos en la columna del rey\n*");

Printf ("*Ruptura en el flanco del rey, se deben preservar los tres peones f2, g2 y h2 para proteger al rey\n*");

En esta última se imprimen también para las casillas homologas en la posición del bando negro y se tiene en cuenta el tipo de enroque (corto o largo) para lanzarlas.

printf("*El rey está protegido por la estructura de peones f2, g2 y h2\n*");

printf("*El peón f2 está siendo atacado por el enemigo\n*");

Printf ("*Existe un déficit de piezas para defender al rey\n*");

Printf ("*El rey amenaza a peón enemigo debilitado* (");

Printf ("*Rey detrás de fila debilitada\n*");

3.1.3 Ventaja espacial y control del centro

Sobre el resto de las piezas se puede percatar de la gran importancia que reviste el control del centro y la ventaja espacial o movilidad en el tablero. Se analiza pieza a pieza del bando a evaluar y se emiten mensajes sobre el control del centro y la movilidad de cada una de ellas.
Printf ("*Control del centro: (%d)* ", n);
Printf ("*Movilidad: (%d)* ", n);

3.1.4 Caballos y Alfiles

Lo discutido anteriormente se tiene en cuenta para todas las piezas de forma general (los peones no son piezas), ahora se explican otros aspectos considerados para cada una por separada, sus características peculiares y se puntualizan las particularidades de la posición de cada pieza que el estudiante debe conocer.

Para poder imprimir los mensajes sobre los caballos, el método recibe como entrada el bando a evaluar como toda la lógica lo indica y se emiten las siguientes sugerencias:
Printf *("%d %s %s* (“, n, cab, clav);
Printf ("*C%s* ", casillas [sq]);
donde en el primer caso se devuelve un aspecto táctico estratégico muy importante que son los caballos clavados, en el cual n es la cantidad. El segundo caso devuelve la casilla ocupada del caballo clavado o de los caballos clavados.

Otras sugerencias para la pieza caballo son:
Printf ("*Caballo defendido por peón\n*");
Printf ("*El caballo amenaza a peón enemigo debilitado* (");
Estas sugerencias también se cumplen para los alfiles, torres (excepto la que trata sobre la pieza defendida por peón) y además están las siguientes para los alfiles en específico:
Printf ("*Alfil en fianchetto\n*");

También se chequean si están atrapados los alfiles para ambos bandos en casillas homólogas, es decir a2-a7.

Printf ("*Alfil a2 atrapado\n*");
Printf *("Alfil h2 atrapado\n");*

3.1.5 Torres y Dama

Para las torres se imprimen las siguientes sugerencias teniendo en cuenta la fase del juego que es calculada por la cantidad de piezas que estén en el tablero, por ejemplo cuando comienza el juego, la variable phase toma valor 1 y en un final 8. Esta es una vía para que el GNU Chess pueda determinar la fase de juego y los consejos al estudiantes tengan una mayor efectividad.
Printf ("*Torre en la columna abierta %c\n*", columnas [fyle]);
Printf ("*Torre en la columna semi-abierta %c\n*", columnas [fyle]);
Printf ("*Torre detrás de peón pasado\n*");
Printf ("*Torre delante de peón pasado\n*");
Printf ("*La torre ubicada en la 7ma fila es muy fuerte contra el rey enemigo\n*");

La dama es la pieza más poderosa y su ataque es devastador por lo que al encontrarse cerca del rey enemigo las posibilidades que tiene de éxito son gigantescas y en particular para ella se imprime la siguiente sugerencia:
Printf ("*La dama cercana al rey enemigo es utilizada para limitar los movimientos del rey contrario\n*");
La forma de implementación es similar a las anteriores piezas recibiendo como parámetro el bando a evaluar.

3.1.6 Final Rey Peón vs. Rey

Uno de los finales más comunes existentes en el juego ciencia, es el de rey y peón contra rey, ahora se analizarán las sugerencias para el bando blanco que serían las mismas para el bando negro sólo cambiándole el sentido al mensaje:
printf *("El peón %s está fuera del alcance del rey enemigo\n*", casillas [sq]*);*
printf *("El rey blanco se encuentra dos filas por delante del peón %s, esto le posibilita ganar la partida\n",* casillas [sq]*);*

printf ("*El rey blanco se encuentra una fila por delante del peón %s\n",* casillas [sq]);
printf ("*El rey blanco en la sexta fila y una fila por delante del peón %s le posibilita ganar la partida a las blancas\n*", casillas [sq]);
printf *("El rey negro se opone al rey blanco\n");*
printf ("*El rey blanco se encuentra en la misma fila que el peón %s\n*", casillas [sq]);
printf ("*El rey negro no está 3 filas por delante del peón %s, el cual está protegido por su rey, posibilitando ganar la partida a las blancas\n*", casillas [sq]);
printf ("*El peón %s se encuentra en la sexta fila y está protegido por su rey, el rey negro no está en oposición al rey blanco, posibilitando ganar la partida a las blancas\n*", casillas [sq]);
printf ("*El peón %s está en la séptima fila y el rey blanco está en la sexta fila*", casillas [sq]);
printf (", *esto provoca que la partida termine en tablas. En caso de tocar el turno a las negras las blancas ganan*", casillas [sq]);
printf (", *de esta forma las blancas ganan. En caso de tocar el turno a las blancas la partida termina en tablas*.", casillas [sq]);
printf ("*El rey negro no bloquea al peón %s\n*", casillas [sq]);
printf ("*El rey negro bloquea al peón %s y ambos reyes están en oposición\n*", casillas [sq]);

3.1.7 Desarrollo de las piezas

Como se explicó anteriormente el desarrollo de las piezas es muy importante, este análisis se realiza solamente en la apertura. Para esto se tiene en cuenta principalmente la cantidad de piezas menores desarrolladas, el enroque y la conexión entre las torres.
printf("*%d pieza menor está desarrollada\n*", n);
printf("*Se ha alcanzado el enroque\n*");
printf("*No se ha alcanzado el enroque\n*");
printf("*Las torres están conectadas\n*");
printf("*Las torres no están conectadas\n*");
printf ("*Movimiento temprano de la dama\n*");
printf ("*Movimiento repetido del caballo %s, en la apertura esto debe evitarse para desarrollar todas las piezas\n*", casillas [sq]);

printf ("*Movimiento repetido del alfil %s, esto debe evitarse en la apertura para un mejor desarrollo de las piezas*\n", casillas [sq]);

3.1.8 Valoración cuantitativa

El análisis que conlleva a determinar la valoración cuantitativa, es decir, qué bando conserva la ventaja es profundo y complejo y se plasma mediante la rutina ImprimeVentaja (). El primer paso es verificar que la posición actual pueda converger a tablas, en caso de serlo se ilustra el siguiente mensaje "*El juego es tablas*" y se obvia continuar evaluando la posición. Si la cuestión es otra entonces se procede a cambiar el modo de juego del GNU Chess a la máxima dificultad para que, y retomando lo expresado en el anterior capitulo, se pueda encontrar la jugada que mejor posición le aporte al motor de ajedrez. Después el modo de juego se asienta como estaba originalmente y se examina el valor del mejor movimiento con un rango predeterminado. Si el valor es menor que 70 se imprime:
printf ("*Posición equilibrada*\n");
De estar entre 70 y 140
printf ("*Ligera ventaja para las* ");
Mayor que 140
printf ("Ventaja superior para las ");
También se ilustra la profundidad de la búsqueda.

3.1.9 Nuevos comandos definidos

La aplicación final debe cumplir el objetivo previamente trazado que es la enseñanza del juego ciencia, debido a esto y retomando lo expresado en el capitulo dos se implementaron dos nuevos comandos para reflejar las sugerencias y ventaja. El comando “sug” se refiere a las sugerencias para lo cual se analiza el bando en turno y se llama a la rutina EvaluaConMensajes () explicada anteriormente. El comando “ventaja” llama a la rutina ImprimeVentaja () también analizada. Ambas contribuyen potencialmente a alcanzar las metas propuestas.

3.2 Aplicación Web

Hasta aquí se ha explicado todo lo relacionado con el motor de juego, las modificaciones que sufrió entre otras cosas, de ahora en lo adelante se hace énfasis en la interfaz de juego, sus características y funcionalidades.

La interfaz de juego es amigable y muy fácil de usar aún por aquellos que no poseen conocimientos informáticos. Antes de comenzar el juego se puede configurar el tablero a una posición determinada presionando el botón LimpiarTablero y arrastrando las piezas deseadas al tablero. Otra vía de configurar el tablero es ir eliminando las piezas que no son necesarias y reorganizar las que aún están en juego hasta llegar a la situación deseada. La solución más completa en este problema de configurar una posición es escribiendo la FEN en un campo de texto para así llegar a una posición determinada. Si en la posición del tablero configurada ya se ha alcanzado el enroque de alguno de los dos bandos se debe marcar el checkbox correspondiente al mismo presentado en la interfaz. A continuación se muestra una imagen de la página Web inicial de la aplicación que se refiere a la modificación del tablero de ajedrez.

Figura. 5. Página inicial de la aplicación.

Una vez presionado el botón Listo se puede jugar escogiendo el bando anhelado. En caso de que la FEN sea inválida se le informa al usuario, en caso contrario se abre la página donde se comenzaría a jugar y aprender. En el nuevo entorno cuando se presiona el botón “JuegaGNU Chess” se le envía la información de la FEN a la página robot.php donde se ejecuta la función JuegaGNUChess ($fen, $nivel, $engine_gnuchess), esta recibe como entrada la FEN, el nivel de juego, siendo este generalmente medio y la dirección donde se encuentra el motor GNU Chess en el servidor. En dicha función se crea el script de entrada del motor y se inicia el proceso del mismo mediante la función StartRobotProcess ($engine_gnuchess, $script), recibiendo por parámetro la dirección del motor y el script de entrada que está conformado por los comandos “setboard” para modificar el tablero a la posición indicada por la FEN recibida, “xboard”, especificando al motor que utilizará dicho protocolo para su comunicación correcta con la aplicación Web, “depth”, donde se define la profundidad de búsqueda del movimiento, “go”, indicándole al motor que realice la jugada y “exit” para finalizar el proceso del motor; posteriormente se interpreta la respuesta del GNU Chess y se visualiza la jugada en el tablero. Cada vez que una persona o el motor realice un movimiento se indica con una flecha desde y hasta dónde se efectúa la jugada. A continuación se ilustra el resultado de dicho proceso.

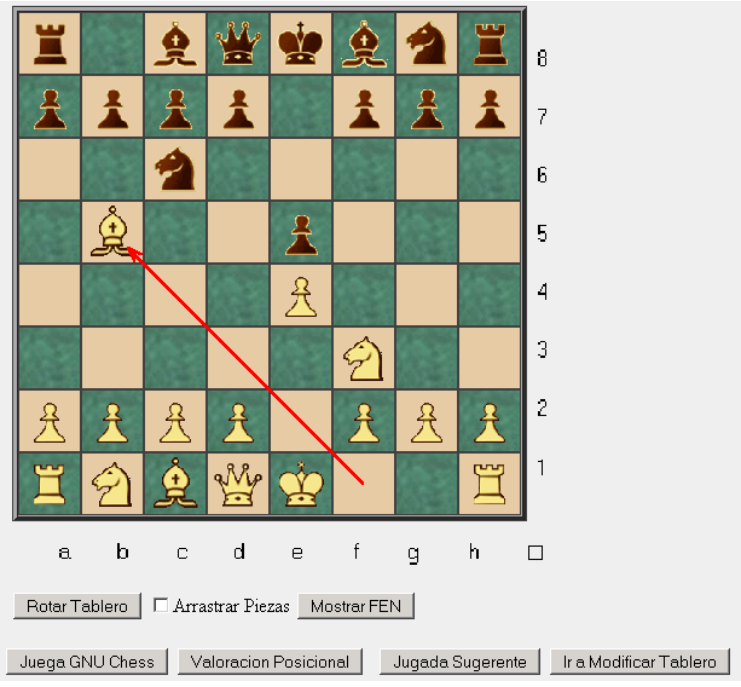

Figura. 6. Proceso de juego.

La interfaz de usuario cuenta con un botón referente a la valoración posicional del tablero donde se muestran las sugerencias y la ventaja de la posición actual descrita anteriormente. El procedimiento interno que se sigue para exponer estos aspectos es prácticamente el mismo que se persigue para que el motor pueda jugar. Las diferencias fundamentales están en que se ejecutan las funciones Sugerencias y Ventaja en vez de JuegaGNUChess ($fen, $nivel, $engine_gnuchess), los parámetros son los mismos. En la página robot.php el script de entrada para la función Sugerencias recibe los comandos setboard, xboard, exit y sug (explicado en este capítulo) mientras que para la función Ventaja el script de entrada tiene como novedad el comando ventaja, ambas llaman a la función StartRobotProcess ($engine_gnuchess, $script) para iniciar el proceso del GNU Chess. También la aplicación cuenta con un botón llamado “Jugada Sugerente” mediante el cual, el usuario puede solicitar otras jugadas válidas y sugestivas. Este es el procedimiento tanto interno como externo que se sigue con la interfaz. A continuación se muestra la forma en que se ilustran las sugerencias al usuario en la aplicación.

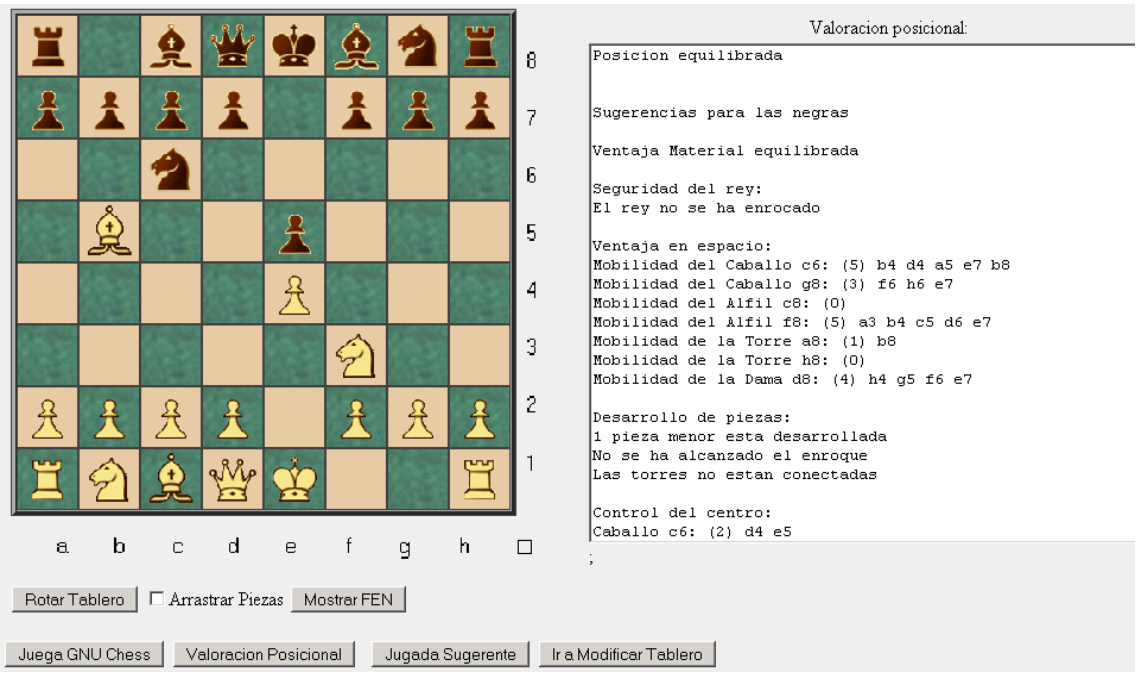

Figura. 7. Valoración Posicional.

3.3 Pruebas del Software

Este epígrafe está dedicado a las pruebas del software para la detección y corrección de los errores que pueda presentar la aplicación con el objetivo de darles solución y respuesta a los mismos, las pruebas no excluyen la aparición de defectos en el software pero permiten analizar la calidad del software con el objetivo de poder efectuar mejoras en futuras versiones de la aplicación. Las pruebas realizadas están dirigidas a las sugerencias que brinda la aplicación al usuario para diferentes posiciones de tableros ya analizadas por maestros del mundo del ajedrez y expuestas en el libro "Ajedrez Integral" de un colectivo de autores del Instituto Superior Latinoamericano de Ajedrez (ISLA).

3.3.1 Prueba 1

La primera prueba se aplicó a la FEN (1r4k1/2qr1pbp/6p1/2pB4/QpPp4/3P2P1/1P3P1P/3RR1K1 w - - 0 1)

Figura. 8. Prueba de Software 1.

Resultado esperado	Resultado Obtenido
• Presión sobre f7 • Peón b2 retrasado • Torre e1 en una columna abierta	✓ Ligera ventaja para las blancas de 73 puntos ✓ Profundidad de búsqueda: 10 • **Sugerencias para las blancas** ✓ Ventaja Material equilibrada ✓ Seguridad del rey: ✓ El rey está enrocado ✓ Ruptura en el flanco del rey, se deben preservar los tres peones f2, g2 y h2 para proteger al rey

• Alfil d5 defendido por peón • Alfil g7 en fianchetto	✓ Estructura de peones: ▪ Peón b2 retrasado ✓ Ventaja en espacio: ▪ Movilidad del Alfil d5: (9) h1 g2 f3 e4 c6 e6 b7 f7 a8 ▪ Movilidad de la Torre d1: (5) a1 b1 c1 f1 d2 ▪ Movilidad de la Torre e1: (11) a1 b1 c1 f1 e2 e3 e4 e5 e6 e7 e8 ▪ Movilidad de la Dama a4: (13) a1 a2 c2 a3 b3 b4 a5 b5 a6 c6 a7 d7 a8 ✓ Alfil d5 defendido por peón ✓ Torre e1 en la columna abierta E • **Sugerencias para las negras** ✓ Ventaja Material equilibrada ✓ Seguridad del rey: ✓ El rey está enrocado ✓ Ruptura en el flanco del rey, se deben preservar los tres peones f7, g7 y h7 para proteger al rey ✓ El peón f7 está siendo atacado por el enemigo ✓ Ventaja en espacio: ▪ Movilidad del Alfil g7: (5) e5 f6 h6 f8 h8 ▪ Movilidad de la Torre d7: (6) d5 d6 a7 b7 e7 d8 ▪ Movilidad de la Torre b8: (8) b5 b6 b7 a8 c8 d8 e8 f8 ▪ Movilidad de la Dama c7: (12) g3 f4 a5 e5 b6 c6 d6 a7 b7 e7 c8 d8 • Alfil g7 en fianchetto

3.3.2 Prueba 2

La segunda prueba se aplico a la FEN
(rr4k1/1b1nbpp1/3p1n1p/2q1p3/1p2P2B/1BN1N3/1PP1QPPP/R2R2K1 w - - 0 1)

Figura. 9. Prueba de Software 2.

Resultado esperado	Resultado Obtenido
• Torre d1 en una columna semi-abierta • Peón b4 retrasado • Peón d6 retrasado • Peón aislado en la columna B • Caballo f6 clavado	✓ Ligera ventaja para las blancas de 128 puntos ✓ Profundidad de búsqueda: 8 • **Sugerencias para las blancas** ✓ Ventaja Material equilibrada ✓ Seguridad del rey: ✓ El rey está enrocado ✓ El rey está protegido por la estructura de peones f2, g2 y h2 ✓ Ventaja en espacio: ▪ Movilidad del Caballo c3: (5) b1 a2 a4 b5 d5 ▪ Movilidad del Caballo e3: (5) f1 c4 g4 d5 f5 ▪ Movilidad del Alfil b3: (6) a2 a4 c4 d5 e6 f7 ▪ Movilidad del Alfil h4: (3) g3 g5 f6 ▪ Movilidad de la Torre a1: (11) b1 c1 e1 f1 a2 a3 a4 a5 a6 a7 a8 ▪ Movilidad de la Torre d1: (9) b1 c1 e1 f1 d2 d3 d4 d5 d6

	▪ Movilidad de la Dama e2: (10) e1 f1 d2 d3 f3 c4 g4 b5 h5 a6 ✓ Torre a1 en la columna abierta A ✓ Torre d1 en la columna semi-abierta D • **Sugerencias para las negras** ✓ Ventaja Material equilibrada ✓ Seguridad del rey: ✓ El rey está enrocado ✓ Ruptura en el flanco del rey, se deben preservar los tres peones f7, g7 y h7 para proteger al rey ✓ El peón f7 está siendo atacado por el enemigo ✓ Estructura de peones: ▪ Peón b4 retrasado ▪ Peón d6 retrasado ▪ Peón aislado en la columna B ✓ Ventaja en espacio: ▪ Movilidad del Caballo f6: (6) e4 g4 d5 h5 h7 e8 ▪ Movilidad del Caballo d7: (2) b6 f8 ▪ Movilidad del Alfil b7: (5) e4 d5 a6 c6 c8 ▪ Movilidad del Alfil e7: (2) d8 f8 ▪ Movilidad de la Torre a8: (11) a1 a2 a3 a4 a5 a6 a7 c8 d8 e8 f8 ▪ Movilidad de la Torre b8: (4) c8 d8 e8 f8 ▪ Movilidad de la Dama c5: (12) c3 e3 c4 d4 a5 b5 d5 b6 c6 a7 c7 c8 ✓ Desarrollo de piezas: ▪ 4 piezas menores están desarrolladas ▪ Se ha alcanzado el enroque ▪ Las torres están conectadas ✓ Control del centro:

	▪ Caballo f6: (2) e4 d5 ▪ Caballo d7: (3) c5 e5 f6 ▪ Alfil b7: (3) e4 d5 c6 ▪ Alfil e7: (2) d6 f6 ▪ Torre a8: (0) ▪ Torre b8: (0) ▪ Dama c5: (8) c3 e3 c4 d4 d5 e5 c6 d6 ✓ 1 caballo clavado (Cf6) ✓ Torre a8 en la columna abierta A

3.3.3 Prueba 3

La tercera prueba se aplicó a la FEN (r4rk1/pp2bppp/3p1n2/4pP2/2q1P3/2N1B3/PPP3PP/R3QRK1 w - - 0 1)

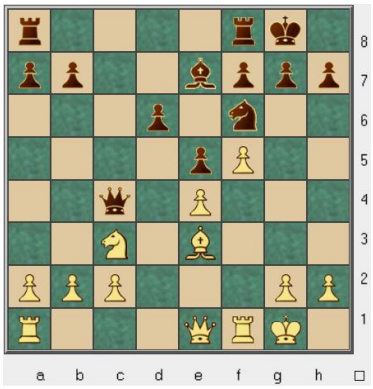

Figura. 10. Prueba de Software 3.

Resultado esperado	Resultado Obtenido
• Peón d6 retrasado • Ventaja de espacio en el	✓ Posición equilibrada • **Sugerencias para las blancas** ✓ Ventaja Material equilibrada ✓ Seguridad del rey:

flanco rey para las blancas	✓ El rey está enrocado ✓ Ruptura en el flanco del rey, se deben preservar los tres peones f2, g2 y h2 para proteger al rey ✓ Ventaja en espacio: ▪ Movilidad del Caballo c3: (6) b1 d1 e2 a4 b5 d5 ▪ Movilidad del Alfil e3: (10) c1 d2 f2 d4 f4 c5 g5 b6 h6 a7 ▪ Movilidad de la Torre a1: (3) b1 c1 d1 ▪ Movilidad de la Torre f1: (6) b1 c1 d1 f2 f3 f4 ▪ Movilidad de la Dama e1: (8) b1 c1 d1 d2 e2 f2 g3 h4 ✓ Desarrollo de piezas: ▪ 2 piezas menores están desarrolladas ▪ Se ha alcanzado el enroque ▪ Las torres están conectadas ✓ Control del centro: ▪ Caballo c3: (2) e4 d5 ▪ Alfil e3: (3) d4 f4 c5 ▪ Torre a1: (0) ▪ Torre f1: (3) f3 f4 f5 ▪ Dama e1: (2) c3 e3 • **Sugerencias para las negras** ✓ Ventaja Material equilibrada ✓ Seguridad del rey: ✓ El rey está enrocado ✓ El rey está protegido por la estructura de peones f7, g7 y h7 ✓ Estructura de peones: ▪ Peón d6 retrasado ✓ Ventaja en espacio: ▪ Movilidad del Caballo f6: (6) e4 g4 d5 h5 d7 e8 ▪ Movilidad del Alfil e7: (1) d8

	▪ Movilidad de la Torre a8: (4) b8 c8 d8 e8 ▪ Movilidad de la Torre f8: (4) b8 c8 d8 e8 ▪ Movilidad de la Dama c4: (18) f1 a2 e2 b3 c3 d3 a4 b4 d4 e4 b5 c5 d5 a6 c6 e6 c7 c8 ✓ Desarrollo de piezas: ▪ 2 piezas menores están desarrolladas ▪ Se ha alcanzado el enroque ▪ Las torres están conectadas ✓ Control del centro: ▪ Caballo f6: (2) e4 d5 ▪ Alfil e7: (2) d6 f6 ▪ Torre a8: (0) ▪ Torre f8: (0) ▪ Dama c4: (8) c3 d3 d4 e4 c5 d5 c6 e6

Las pruebas realizadas a la aplicación reflejan el correcto funcionamiento del software en los distintos casos analizados, aunque no se puede asegurar cien por ciento su desempeño sin incertidumbre alguna porque en el juego del ajedrez pueden surgir disímiles de escenarios complejos de evaluar. Se considera que la aplicación cumple con las condiciones para funcionar correctamente durante el proceso de enseñanza del ajedrez.

CONCLUSIONES

En este trabajo se ha mostrado la posibilidad de utilizar técnicas de Inteligencia Artificial en la enseñanza del ajedrez, donde la toma de decisiones desempeña un papel importante. Se ha prestado particular interés en la enseñanza del juego ciencia, empleando como base de conocimiento fundamental un programa o motor de ajedrez. El núcleo del trabajo se ha centrado en el rediseño del motor GNU Chess para sugerir información al usuario que lo apoye de forma más interactiva en el aprendizaje del juego, mostrando una aplicación web donde el usuario puede modificar la posición del tablero a partir de la cual desea jugar y analizar aspectos fundamentales que se tienen en cuenta en la enseñanza del juego ciencia. Precisamente este es el objetivo fundamental del trabajo y sin duda alguna constituye una novedad en este tipo de investigación.

RECOMENDACIONES

- Dar seguimiento a este trabajo para obtener un mayor y mejor resultado del obtenido hasta ahora, esto significa que para una próxima iteración la aplicación no solo permita al estudiante aprender a jugar ajedrez a un nivel básico logrando un mayor nivel de abstracción.
- Mejorar el sistema de evaluación del motor GNU Chess para obtener un nivel superior.
- Utilizar la aplicación obtenida en la cátedra de ajedrez de la Universidad de las Ciencias Informáticas en el proceso de enseñanza-aprendizaje.

REFERENCIAS BIBLIOGRÁFICAS

1. Comité Técnico de Arbitros de la FEDA. LEYES DEL AJEDREZ DE LA FIDE. Julio 2005. http://www.feda.org/leyes/leyes.pdf.

2. Rusell, S. J. y Norvig, P. *Inteligencia Artificial. Un Enfoque Moderno. Segunda ed.* Madrid : Pearson Educación S.A., 2004.

3. M., Jorge Egger. DESARROLLO DE LA MAESTRÍA AJEDRECÍSTICA COMPUTACIONAL. Julio 2003. http://www.fenach.cl/docs/memoria/node12.html.

4. Hyatt, Robert. Computer Chess (Crafty). 2006. http://www.cis.uab.edu/info/faculty/hyatt/hyatt.html.

5. Chessville. 50 Chess Games for Beginners. Febrero 9, 2003. http://www.chessville.com/links/Site%20Reviews/50_Chess_Games_for_Beginners.htm.

6. EL PORTAL DE AJEDREZ. SOFTWARE DE AJEDREZ. 2008. http://www.schackportalen.nu/Espanol/esmjukvara.htm.

7. Chessbase. El pequeño Fritz. 2005. http://www.chessbase.com/espanola/shop/product.asp?pid=170.

8. Iinforchess. TOTAL CHESS TRAINING. 2006. http://www.inforchess.com/catalogo/TCT.htm.

9. SHANNON, CLAUDE E. XXII. Programming a Computer for Playing Chess. 1950. http://archive.computerhistory.org/projects/chess/related_materials/text/2-0%20and%202-1.Programming_a_computer_for_playing_chess.shannon/2-0%20and%202-1.Programming_a_computer_for_playing_chess.shannon.062303002.pdf.

10. Ajedrezmagico. Notación PGN. Marzo 2008. http://ajedrezmagico.blogspot.com/2008/03/notacion-pgn.html.

11. Ajedrezmagico. Notación FEN. Febrero 2008. http://ajedrezmagico.blogspot.com/2008/02/notacion-fen.html.

12. Mitchell, T. *Machine Learning, McGraw Hill.* 1997.

13. Ibañez, Carlos y Fuentes, Carolina. Algoritmos más utilizados en Aprendizaje Automático. 2007. http://www.comenius.usach.cl/gvillarr/cursoia/alumnos/fuentesibanez/links%20rellacionado/algoritmo_aa.html.

14. Ibañez, Carlos y Fuentes, Carolina. Árboles de decisión. 2007. http://www.comenius.usach.cl/gvillarr/cursoia/alumnos/fuentesibanez/links%20rellacionado/arboles_aa.html.

15. Ginits, Herbert. *Game Theory Evolving.* s.l. : Princeton University Press, 2000.

16. Tompson, Raúl Garreta. Un jugador de Go basado en Aprendizaje Automático. 2006. http://www.fing.edu.uy/~pgagrego/.

17. Ocaña, A.M. *Juegos como problemas de Búsqueda. Búsqueda con adversarios y grafos.* 2007.

18. Rebaza, Jorge Valverde y Juárez, Pedro Shiguihara. *Algoritmo de búsqueda Minimax con poda Alfa/Beta y búsqueda Quiescense para el juego del Ajedrez.* 2008.

BIBLIOGRAFÍA CONSULTADA

1. Ajedrezmagico. Notación PGN. Marzo 2008. http://ajedrezmagico.blogspot.com/2008/03/notacion-pgn.html.
2. Bernal, Jesus Antonio González. Aprendizaje Computacional. 2009. http://ccc.inaoep.mx/%7Ejagonzalez/ML/principal/principal.html.
3. Buckland, Mat. *Programming Game AI by Example.* 2005.
4. Colectivo de autores del Instituto Superior Latinoamericano de Ajedrez (ISLA). *Ajedrez Integral, tomo I.* La Habana, 2003.
5. David N. L. Levy y Newborn, Monty. *How Computers Play Chess.* 2009.
6. Defez Gómez, Juan Francisco. La Informática aplicada en Ajedrez. 1999. http://www.inforchess.com/compulib/defez/defez01.htm.
7. Ginits, Herbert. *Game Theory Evolving.* s.l. : Princeton University Press, 2000.
8. J.D, Funge. *Artificial Intelligence for Computer Games.* 2004.
9. Lezcano, B. M. *Introducción a la Inteligencia Artificial. ¿Qué es la IA?* s.l. : Pandora, 2000.
10. M, Gómez. *Aplicaciones de la IA.* s.l. : Ediciones de la Noche, 2002.
11. M, Jorge Egger. DESARROLLO DE LA MAESTRÍA AJEDRECÍSTICA COMPUTACIONAL. Julio 2003. http://www.fenach.cl/docs/memoria/node12.html.
12. Mitchell, T. Machine Learning, McGraw Hill. 1997.
13. Ocaña, A. M. Juegos como problemas de Búsqueda. Búsqueda con adversarios y grafos. 2007.
14. Pompa, Alexander Zardina. *Aplicación del algoritmo Min-Max en el desarrollo de juegos de contrincantes en un entorno virtual.* La Habana. Universidad de las Ciencias Informáticas, 2008.
15. Rebaza, Jorge Valverde y Juárez, Pedro Shiguihara. *Algoritmo de búsqueda Minimax con poda Alfa/Beta y búsqueda Quiescense para el juego del Ajedrez.* 2008.
16. Rusell, S. J. y Norvig, P. *Inteligencia Artificial. Un Enfoque Moderno. Segunda ed.* Madrid : Pearson Educación S.A., 2004.
17. SHANNON, CLAUDE E. XXII. Programming a Computer for Playing Chess. 1950. http://archive.computerhistory.org/projects/chess/related_materials/text/2-0%20and%202-1.Programming_a_computer_for_playing_chess.shannon/2-0%20and%202-1.Programming_a_computer_for_playing_chess.shannon.062303002.pdf.
18. Tompson, Raúl Garreta. Un jugador de Go basado en Aprendizaje Automático. 2006. http://www.fing.edu.uy/~pgagrego/.
19. Universidad de las Ciencias Informáticas. Entorno Virtual de Aprendizaje. 2009. http://eva.uci.cu.
20. Free Software Foundation Inc. GNU Chess. Agosto 2003. http://www.gnu.org/software/chess/chess_faq.html#B.1.

ANEXOS

Anexo # 1

#	Name	ELO
1	Rybka 3 64-bit	3155
2	Naum 4 64-bit	3055
3	Thinker 5.4C Inert 64-bit	2969
4	Zappa Mexico II 64-bit	2959
5	Fritz 11	2958
6	Grapefruit 1.0 32-bit	2945
7	Deep Shredder 11 64-bit 1CPU	2940
8	Deep Sjeng WC2008 64-bit 1CPU	2938
9	Hiarcs 12.1	2919
10	Onno 0.12.0 64-bit	2890
11	Loop 13.6 32-bit	2882
12	Glaurung 2.2 64-bit	2874

Figura 11. Mejores motores de ajedrez (mayo del 2009) para un solo procesador.

Anexo # 2

No.	Año	Lugar	Ganador
1	1974	Stockholm	Kaissa
2	1977	Toronto	Chess 4.6
3	1980	Linz	Belle
4	1983	New York, NY	Cray Blitz
5	1986	Cologne	Cray Blitz
6	1989	Edmonton, Canada	Deep Thought
7	1992	Madrid, Spain	ChessMachine (Gideon)
8	1995	Hong Kong	Fritz
9	1999	Paderborn, Germany	Shredder
10	2002	Maastricht, Netherlands	Deep Junior
11	2003	Graz, Austria	Shredder
12	2004	Bar-Ilan University, Ramat Gan, Israel	Deep Junior
13	2005	Reykjavík, Iceland	Zappa
14	2006	Torino, Italy	Junior
15	2007	Amsterdam, The Netherlands	Rybka
16	2008	Beijing, China	Rybka
17	2009	Pamplona, Spain	Rybka

Tabla 1. Ganadores de los campeonatos mundiales de programas de ajedrez.

GLOSARIO

A

Abandonar: Opción que tiene el jugador de ajedrez, implica la pérdida de la partida.

Alfa-Beta: Variante del algoritmo Minimax que no explora nodos innecesarios**.**

Algoritmo: Conjunto de reglas bien definidas para la solución de un problema en un número finito de pasos.

Ahogo: Posición que conduce al resultado de tablas o empate. Implica ausencia total de movimiento para un bando o color sin que el Rey esté en Jaque.

Amenaza: En ajedrez, situación latente que implica peligro para uno de los dos bandos, blancas o negras.

Aprendizaje Automático: Rama de la Inteligencia Artificial cuyo objetivo es desarrollar técnicas que permitan a las computadoras aprender. Se trata de crear programas capaces de generalizar comportamientos a partir de una información no estructurada suministrada en forma de ejemplos. Es, por lo tanto, un proceso de inducción del conocimiento.

Ataque: En ajedrez, está íntimamente relacionado con la iniciativa, consiste en situar piezas propias en relación con las del adversario, de tal forma que este se vea obligado a defenderse. El ataque es efectivo cuando obliga al adversario a jugar a la defensiva en lugar de realizar jugadas constructivas para sus planes.

Autómata: Equipo electrónico programable en lenguaje no informático y diseñado para controlar, en tiempo real y en ambiente industrial, procesos secuenciales.

B

Bash: es un shell de Unix (intérprete de órdenes de Unix) escrito para el proyecto GNU

Búsqueda Exhaustiva: Búsqueda completa en todo el árbol de juego.

C

Cambio: En ajedrez, se produce cuando desaparecen en un mismo cuadro piezas de ambos bandos, es decir, ocurre una captura en un cuadro por parte de un bando y en ese mismo cuadro, el otro bando también captura. El dominio de los cambios en el ajedrez es muy importante; cuando cambiar y cuando no, que elementos cambiar y cuál es su valor, son aspectos que pueden revelar un grado de maestría en el dominio del juego.

Captura o toma: En ajedrez, eliminar un elemento o pieza del adversario, este desaparece del tablero.

Centro: En ajedrez, área central del tablero conformada por los cuadros e4, e5, d4, d5. Es muy importante porque en ella las piezas aumentan sus posibilidades dinámicas, tanto en número de cuadros como en direcciones de movimiento.

Columna: En ajedrez, existen en número de ocho en el tablero y están formadas por el conjunto de cuadros ubicados en posición vertical dentro del mismo. Ofrecen posibilidades de movimiento a la Dama y a las Torres y pueden estar en tres estados: abiertas (ausencia total de peones en ella); semiabiertas (cuando en ella existe un Peón de uno de los dos bandos) y cerradas (cuando existen peones de ambos bandos en ella).

Cuadro: En ajedrez, también llamado casilla, escaque o punto, existen en número de 64 en el tablero de ajedrez y constituyen la unidad básica del mismo. Pueden estar en dos estados; libres u ocupados por las piezas y/o peones.

D

Debilidad: En ajedrez, situación que se produce sobre la base de las dificultades que cada uno de los bandos tiene en la posición. Las debilidades pueden ser temporales o tener un carácter más permanente, por ejemplo una pieza indefensa o mal coordinada con el resto puede ser una debilidad temporal, pero un peón aislado es una debilidad más permanente.

E

Elementos operativos: En ajedrez, puede denominarse así a todas las piezas y los peones que constituyen la Fuerza o Material en el ajedrez y se desplazan por el tablero ocupando sus cuadros o puntos. Los elementos operativos, junto con el Espació o tablero y el Tiempo u oportunidad de jugar, conforman la Posición.

Enroque: Movimiento especial en el ajedrez que se realiza para proteger el rey y sólo se puede hacer cuando este aún no se ha movido, ni está siendo amenazado en el momento de ejecutar dicho movimiento. Consiste en mover el rey dos casillas en dirección de la torre y esta última se posiciona en la casilla del lado opuesto en que se movió el rey.

F

Fianchetto: Término italiano con que se denomina en el ajedrez al desarrollo diagonal del Alfil en la Apertura, ocupando los cuadros g2 o b3 para las blancas y g7 y b7 para las negras, por ejemplo: 1.g3 g6 2.Ag2 Ag7; en estas jugadas de blancas y negras ambos bandos han efectuado un fianchetto.
Fila: En ajedrez, Existen en número de ocho en el tablero y están constituidos por un conjunto de cuadros en posición horizontal. Por ellas se desplazan la Dama, las Torres y el Rey de un flanco a otro del tablero.
Flanco: Al trazar una línea entre las columnas d y e, se forman los flancos del tablero de ajedrez en los que las piezas de ambos bandos queda distribuidas de forma simétrica con la excepción de la dama y el rey. Es por eso por lo que el lado de la dama se denomina flanco de dama, y el del rey, flanco de rey.
Framework: Es una estructura de soporte definida, mediante la cual otro proyecto de software puede ser organizado y desarrollado. Típicamente, puede incluir soporte de programas, bibliotecas y un lenguaje interpretado entre otros software para ayudar a desarrollar y unir los diferentes componentes de un proyecto.
Freeware: Define un tipo de software de computadora que se distribuye sin coste y por tiempo ilimitado

G

Gambito: En ajedrez, sacrificio o entrega de material en la Apertura con un objetivo determinado, el más frecuente de ellos dificultar el desarrollo del adversario y acelerar el desarrollo propio. Cuando lo hacen las negras muchas veces se le denomina contragambito.
GNU Chess: Programa de ordenador, creado para jugar ajedrez.

H

Hashtype: perteneciente a la tabla de transposición
Heurística: Regla que permite orientar un algoritmo hacia la solución de un problema. Técnica de programación que permite a un sistema la creación gradual de un valor óptimo para una variable específica por medio del registro de los valores obtenidos en operaciones anteriores. Técnica empleada en los sistemas de inteligencia artificial.

I

Igualdad: En ajedrez, situación equilibrada de la posición que implica que no haya ventaja para ningún bando.

Iniciativa: En ajedrez, lleva consigo la facultad de hostigar al enemigo u obligarle a jugar de cierta manera, es decir, se va desarrollando en una serie de amenazas y ataques de las que el adversario se tiene que defender cediendo terreno en la lucha y debilitándose.

Inteligencia Artificial (IA): La IA dentro de las ciencias de la computación es la encargada de la creación de maquinas que implementan tareas relacionadas con el comportamiento humano, logrando gran similitud al pensamiento del hombre.

J

Jaque: En ajedrez, amenaza al Rey, que obligatoriamente hay que evadir en la jugada correspondiente, esto puede hacerse moviendo el Rey, cubriéndolo con alguna pieza o Peón o capturando al elemento que lo amenaza.

Jaque Mate: En ajedrez, posición final que constituye el objetivo del juego. En ella un Rey está amenazado y en la jugada inmediata no puede escapar de ninguna forma a esta amenaza, por tanto un bando gana y el otro pierde la partida.

Jugada: Movimiento de un elemento operativo, pieza o peón, en ella se verifica el tiempo en el ajedrez. El derecho a jugar pasa alternativamente, en proporción 1 a 1, de un jugador a otro.

M

Minimax: Es un método de decisión para minimizar la pérdida máxima esperada en juegos con adversario y con información perfecta.

Motor: Se refiere a un programa que realiza cierta funcionalidad, en el presente trabajo es mencionado motor al programa de ajedrez GNU Chess.

N

Notación: Registro de la partida de ajedrez. Existen los llamados Sistemas de Notación Algebraico y Descriptivo, siendo el oficial de la FIDE desde 1980, el Algebraico por su mayor sencillez y precisión.

O

Open source: código abierto o libre

P

Promoción: En ajedrez, llegada de un peón a la última fila del tablero donde tiene que ser cambiado por una pieza cualquiera (menos el Rey), sin importar si aún permanece en el tablero, Por tanto se pueden tener dos Damas o tres Caballos etc.

S

Script: Es un programa que puede acompañar a un documento HTML o que puede estar incluido en él. El programa se ejecuta en la máquina del cliente cuando se carga el documento, o en algún otro instante, como por ejemplo cuando se activa un vínculo.

T

Tablas: Resultado de empate en el ajedrez al cual puede llegarse por diversas vías. Cuando ocurre en una competencia el punto o unidad en discusión en la partida se divide a la mitad; 0,5 para cada contendiente.

Tablero: Escenario de la partida de ajedrez (Espacio); está cuadriculado con medidas de 8 x 8 cuadros, los cuales tienen alternativamente colores claros y oscuros.

U

Usuario: Individuo o persona que interactúa con un programa de computación.

V

Ventaja: En ajedrez, determina la superioridad de un bando sobre otro. Cada contendiente lucha por obtener ventaja y el logro de la victoria está dado por el incremento de la misma.