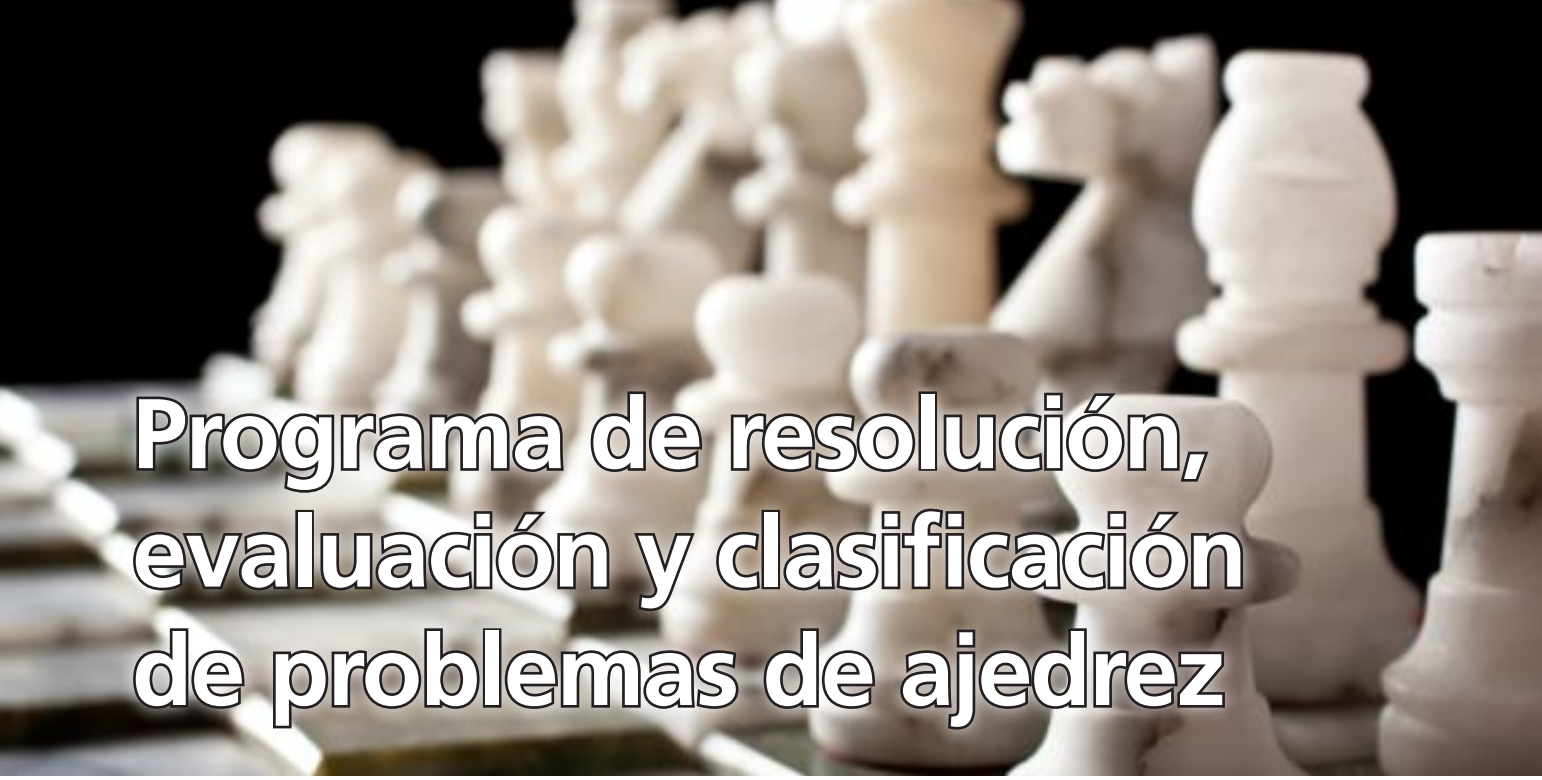


Programa de resolución, evaluación y clasificación de problemas de ajedrez

Palabras clave: Algoritmo de movimientos, desarrollo y resolución de problemas de ajedrez, algoritmo basado en puntuaciones, hostigar pieza, valoración de jugadas, defender una posición.

Resumen:

El proyecto consiste básicamente en realizar una aplicación que permita el estudio de partidas avanzadas de ajedrez, es decir, a partir de una situación dada en un tablero, poder estudiar los diferentes movimientos que puede hacer tanto el usuario (o jugador) como el ordenador que representa al oponente. El usuario podrá elegir qué figuras podrá mover el oponente (para reducir la lista de movimientos posibles); además, el usuario podrá valorar jugadas para que, cuando reproduzca de nuevo la partida, aparezca un comentario con el movimiento que ha realizado. El usuario contará con una serie de mecanismos para conseguir guardar todos los progresos conseguidos con el uso de la aplicación, así como mecanismos para recuperar dichos progresos cuando él desee. Por último, la aplicación contará con un pequeño algoritmo que proporcione una ayuda al usuario a la hora de decidir cuál es el siguiente movimiento a realizar y recomendará al usuario aquellos movimientos que el programa considere mejores dada una situación de partida.

Key words: *Algorithm movement, development and solving chess problems, score-based algorithm, harassing-piece, valuation of hands, defending a position.*

Abstract:

The project is basically to make an application that allows the study of advanced chess games from a given situation on a board, studying different movements that can make both: the user (player) and the computer that represents the opponent. The user can choose which pieces can move the opponent (to reduce the list of possible movements) and value each movement, when the game is reproduced again, thanks to the comment showed. The user will have a number of mechanisms to get to keep all the progress achieved by the use of implementation and mechanisms to recover such progress when he wants. Finally, the application will have a small algorithm to provide assistance to the user to decide the next move to do, recommending to the user movements that the program consider better according to one situation during the game.

Javier Luciano Peña

Estudiante de 5º curso de Ingeniería Informática y 2º curso de Ingeniería en Organización Industrial, en la Universidad Pontificia Comillas. Prácticas en 2010 en Caja Madrid.

Introducción

La idea del desarrollo de una computadora capaz de jugar al ajedrez empezó a difundirse en el siglo XVIII.

En el año 1769 se desarrolló el primer autómata capaz de jugar al ajedrez; este autómata se llamaba "El turco" y resultó ser un engaño, ya que según como estaba diseñado, permitía a una persona introducirse dentro de él para jugar la partida, haciendo creer a los presentes que el autómata funcionaba.

En 1912 el español Leonardo Torres y Quevedo construyó un autómata capaz de jugar al ajedrez, llamado "El Ajedrecista". El autómata hacía uso de electroimanes bajo un tablero de ajedrez y jugaba automáticamente un final de rey y torre contra el rey de un oponente humano. No jugaba de manera muy precisa y no siempre llegaba al mate en el número mínimo de movimientos a causa del algoritmo simple que evaluaba las posiciones. Pero sí lograba la victoria en todas las ocasiones.

Fue el primer autómata que podía jugar sin intervención humana.

Tras estos intentos de crear la primera computadora para poder jugar al ajedrez, la idea cayó en el olvido hasta los años 50.

En los años 50, debido al avance tecnológico y al nuevo impulso que recibió el ajedrez, se empezaron a desarrollar nuevas máquinas capaces de desarrollar partidas contra humanos.

Antes del desarrollo de la primera máquina de ajedrez de los años 50, el ingeniero Claude Elwood Shannon escribió sobre dos tipos de algoritmos para programar el autómata. Estos tipos de algoritmos eran:

- Algoritmos tipo A: el algoritmo se basaba en una búsqueda por fuerza bruta, es decir, el autómata debía examinar todas las posibilidades de movimiento que existían y elegir entre ellas cual era la mejor opción de movimiento (resultaba inviable debido a la cantidad de opciones que había que estudiar)
- Algoritmos tipo B: este algoritmo usaría una especie de "inteligencia artificial estratégica" para solucionar estos problemas en los que únicamente se analizarían sólo 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.

Descripción del proyecto

El trabajo del proyecto consta de tres partes claramente diferenciadas, que se van a describir en los siguientes apartados:

Interfaz de la aplicación:

La interfaz de la aplicación es lo primero que se ha desarrollado del proyecto debido a la necesidad de comprobar el aspecto externo de la aplicación antes de añadirle funcionalidad y dar una idea, tanto al desarrollador como al usuario, de cuál será el aspecto final de la aplicación.

Todo el desarrollo del interfaz se ha realizado en java, debido a:

- Facilidad de programar interfaces.
- Java al programar con orientación a objetos da la posibilidad de usar objetos y clases en sus desarrollos con lo que será más fácil programar objetos y clases.
- Facilidad a la hora de comprobar los resultados obtenidos.
- Portabilidad de la aplicación para distintos sistemas operativos.

El autor ha intentado realizar un interfaz sencillo y fácil de manejar, alejado de las dificultades propuestas por desarrollos con interfaces demasiado vistosos, con lo que se ha buscado la sencillez y la limpieza a la hora de realizar la aplicación.

El interfaz cuenta con una ventana principal (Figura 1) desde donde se podrán realizar todas las operaciones básicas de la aplicación. Esto aportará sencillez y el usuario no se perderá navegando en sub-ventanas que dificulten el uso de la aplicación.

A la ventana principal le apoyarán un conjunto de sub-ventanas para realizar una serie de operaciones básicas, lo que permitirá descargar a la ventana principal de contenido, haciéndola más utilizable y comprensible para el usuario. Este conjunto se sub-ventanas están definidas más adelante en la memoria (ver capítulo 6 apartado diseño externo, epígrafe sub-ventanas).

Desarrollo del sistema experto:

Para conseguir calcular los movimientos de las piezas en el programa y conseguir ver todos los posibles movimientos que se pueden realizar en una situación dada, se ha tenido que crear un sistema experto capaz de calcular los movimientos

Figura 1.

de las figuras en juego, cumpliendo una serie de restricciones que marca el juego del ajedrez.

Para ello se ha desarrollado todo un sistema experto conectado al interfaz para que, una vez comenzada la partida a partir de una serie de figuras, y viendo a quien le toca mover, se calculen todos los movimientos que puedan realizar las piezas dada esa situación.

Se han tenido que crear una serie de funciones para cada tipo de pieza, ya que cada una de las piezas del ajedrez puede realizar diversos movimientos. Una vez que se sabe la pieza a mover y conociendo la posición de las demás piezas en el tablero, se calcularán los posibles movimientos que puede realizar dicha pieza.

Una vez calculados los movimientos que pueden realizar todas las piezas, se devolverá la información al interfaz para que muestre los resultados y de la opción al usuario de elegir el movimiento que desee realizar.

Todo el sistema experto se ha desarrollado en java, ya que permite encapsular los objetos y resulta más fácil comprobar y chequear todas las restricciones que debe cumplir un movimiento antes de añadirlo a la posible lista de movimientos a realizar.

El sistema experto se ha implantado de fondo en la aplicación y el uso de dicho sistema experto es totalmente transparente para el usuario, ya que el usuario no sabe cuando el sistema experto está funcionando para calcular los movimientos.

Algoritmo de localización de movimientos:

Una vez que el usuario está estudiando la partida, y ante la posibilidad de realizar un movimiento dada una situación, la aplicación puede recomendar al usuario realizar unos movimientos que, a priori y bajo unas condiciones marcadas por el autor del proyecto, pueden ser mejores para la situación dada.

Se realizarán dos algoritmos diferentes:

• Algoritmo básico: todas las acciones contendrán los mismos pesos y se bajará un solo nivel de profundidad.

• Algoritmo avanzado: se bajará dos niveles de profundidad y se dotará a las acciones de diferentes pesos.

El algoritmo que se va a realizar en esta aplicación se va a basar en puntuaciones.

Cada figura contendrá un valor según su peso en el juego, este valor se puede apreciar en la Tabla 1.

Como se puede observar, la pieza con mayor valor es el rey, ya que el rey es la pieza en torno a la que gira toda la partida de ajedrez.

Las demás piezas están valoradas según la cantidad de espacio que pueden ocupar en el tablero de ajedrez con sus movimientos, por lo que la reina es la segunda pieza más valiosa (debido a la cantidad de espacio que puede llegar a ocupar en el tablero con sus movimientos) mientras que el peón es la pieza menos valiosa (debido al poco espacio que ocupa en el tablero a la hora de realizar un movimiento).

Una vez marcado el valor de las piezas habrá que valorar las posibles acciones que pueden realizarse en la partida. Estas acciones podrán ser:

• Positivas:
  – Colocar una pieza bajo protección de otra
  – Comer una pieza contraria
  – Amenazar a otra pieza
  – Conseguir proteger a otra pieza desprotegida
• Negativas:
  – Desproteger una pieza
  – Exponer la pieza a ser comida
  – Desproteger la pieza movida

Una vez que se cuenta con ambas valoraciones la fórmula a seguir será la siguiente:

Valor movimiento =∑valores positivos -∑valores negativos

Valor negativo = valor acción negativa*valor figura implicada

Valor positivo = valor acción positiva*valor figura implicada

Valor negativo =valor acción negativa*valor figura implicada

En el caso del algoritmo avanzado, lo que se va hacer, aparte de bajar un nivel más de profundidad en el árbol creado, es dotar a las acciones negativas y a las positivas de un valor diferente al que se le da en el algoritmo básico.

Se les ha dado nuevos valores por las siguientes razones:

• Sería irreal valorar las acciones con el mismo peso, ya que hay algunas que son más importantes que otras.

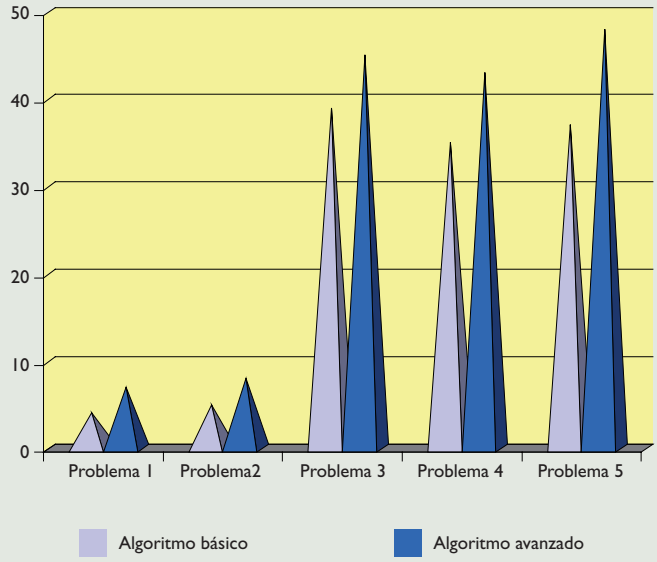
• En lo que respecta a las acciones a favor o en contra, hay que tener en cuenta que se valoran igual acciones que van a tener que realizarse en un periodo de tiempo más largo que otras que se pueden realizar en un periodo de tiempo menor.

Las valoraciones aportadas a las posibles acciones son:

Tabla 1.

Tipo de figura	Valor de la figura
Peón	**1**
Caballo	**3**
Arfil	**3.5**
Torre	**5.5**
Reina	**10**
Rey	**10000**

Figura 2. Movimientos coincidentes entre el algoritmo y "Fritz"



- Acciones positivas:
  - Colocar una pieza bajo protección de otra (valoración 1)
  - Comer una pieza contraria (valoración 2)
  - Amenazar a otra pieza (valoración 0.5)
  - Conseguir proteger a otra pieza desprotegida (valoración 1)
- Acciones negativas:
  - Desproteger una pieza (valoración 1.5)
  - Exponer la pieza a ser comida (valoración 2)
  - Desproteger la pieza movida (valoración 1)

Resultados:

Conclusiones:

• Se observan dos tendencias completamente diferentes: los problemas 1 y 2 siguen una tendencia totalmente diferente a la que siguen los problemas 3, 4 y 5, por lo que se puede comprobar que el algoritmo, dependiendo de qué tipo de problema sea, de apertura o de cierre, obtendrá resultados totalmente distintos.

• Cuando los problemas son de apertura (3, 4 y 5), los resultados que aporta el algoritmo son bastante válidos y ayudan al usuario a la hora de tomar una decisión, ya que es más fácil decidir un movimiento cuanto hay muchas piezas en juego.

• Cuando los problemas son de cierre (1 y 2), los resultados que aporta el algoritmo al usuario no son útiles ya que muy pocas veces acierta el algoritmo con el movimiento que realiza el programa "Fritz", ya que al contar con pocas piezas el algoritmo no puede definir con precisión qué movimientos son mejores.

• Se observa en la Figura 1, que el algoritmo avanzado obtiene siempre mejores resultados que el algoritmo básico, por lo que se deduce que el algoritmo avanzado es un poco más fiable que el algoritmo básico. Esta mejora no es muy apreciable ya que ambos algoritmos obtienen muy buenos resultados o muy malos.

• En la Figura 2 se observa que el algoritmo básico tiene una tasa de fallo mayor que el algoritmo avanzado, por lo que el algoritmo básico es menos fiable que el algoritmo avanzado aunque esta diferencia es muy baja.

Tras analizar los resultados obtenidos y realizar unas conclusiones sobre los mismos, falta añadir una conclusión general sobre los algoritmos.

Lo que se puede concluir es que el algoritmo es una herramienta de ayuda cuando la partida está comenzando o hay muchas piezas en juego, ya que los resultados aportados por el algoritmo se parecen mucho a las decisiones tomadas por el programa "Fritz", pero en caso de que el problema cuente con pocas piezas o la partida esté a punto de terminar, el algoritmo no sirve de ayuda al usuario. ∎

Figura 3.