Práctica 1. Motor de juego de Conecta 6

Félix Escalona Moncholí
Ciencia de la Computación e Inteligencia Artificial. Universidad de Alicante.

Septiembre, 2023

1 Introducción

En esta práctica el objetivo es implementar un motor de juego para el juego Conecta 6.

Para ello, se utilizarán los algoritmos de búsqueda adversaria en grafos vistos en clase, así como algunas de las optimizaciones sugeridas para conseguir que funcione con limitación de tiempo.

En las siguientes secciones se explicarán las reglas del juego así como los requisitos que debe cumplir el motor de juego desarrollado.

2 Conecta 6

Conecta 6 es un juego de estrategia para dos jugadores. En él, dos jugadores, negras y blancas, colocan alternativamente dos piedras de su propio color en las intersecciones vacías de un tablero similar al del Go, excepto que el negro (el primer jugador) coloca únicamente una piedra en su primer movimiento. El primero en conseguir 6 o más piedras consecutivas (horizontalmente, verticalmente o diagonalmente) gana la partida.

Las reglas del Conecta6 son muy simples:

- Jugadores y piedras. Hay dos jugadores. Las negras juegan primero, y las blancas responden. Cada jugador juega con el color de piedras correspondiente.
- Tablero de juego. Se juega en un tablero cuadrado compuesto de líneas ortogonales, con cada intersección pudiendo contener una piedra. En teoría, el tablero puede ser de cualquier tamaño, aunque el tablero de Go de 19x19 se considera el más conveniente.
- Movimientos del juego. Las negras juegan primero, colocando una piedra negra en la intersección. De manera subsiguiente, las blancas y las negras

colocan dos piedras en casillas no ocupadas en cada turno de manera alternativa.

- Ganador. El primer jugador en conseguir 6 o más piedras en fila (horizontalmente, verticalmente o diagonalmente) gana.

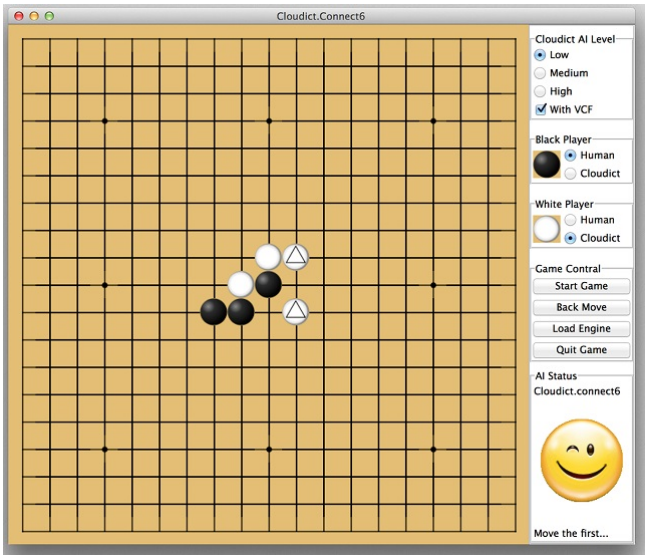

Figure 1: Tablero con una posición de muestra del juego Conecta 6.

3 Motor de juego

Se debe programar un motor de juego que sea capaz de jugar a Conecta6 con ambos colores y a partir de cualquier posición predeterminada.

En primer lugar, se deberá programar un algoritmo de búsqueda utilizando alguna de las técnicas vistas en teoría (o alguna otra alternativa debidamente justificada). A esta búsqueda se le deberán añadir optimizaciones que poden el árbol de variantes de forma que se pueda dar una respuesta en un tiempo acotado, ya que el objetivo es que sea utilizado en tiempo real contra otros programas o jugadores humanos.

Para evaluar las posiciones, se deberá crear una función de evaluación que permita detectar las condiciones de finalización del juego, así como heurísticas que permitan aproximar la evaluación estática de cualquier posición. La ponderación de las heurísticas se podrá hacer de manera manual o utilizando algoritmos de aprendizaje automático (ver Sección 4).

Todo el programa deberá poderse ejecutar en CPU, y se tendrá en cuenta el nivel de optimización del algoritmo utilizado a nivel de consumo de CPU y RAM.

4 Heurística entrenada con algoritmos de aprendizaje automático

La ponderación de distintas heurísticas en la función de evaluación estática de la posición es una tarea muy complicada. Para ello, en el temario se han explicado varias técnicas de aprendizaje automático para permitir la optimización de los pesos de cada una de las características, como es el caso de los algoritmos genéticos.

En esta práctica se propone utilizar alguna alternativa de aprendizaje automático para mejorar la precisión de la función de evaluación estática, ya sea optimizando los pesos de las características diseñadas manualmente, o bien aprendiendo automáticamente dichas características a partir de la posición y la función de evaluación correspondiente.

5 Interfaz del motor de juego

Con el objetivo de permitir la comunicación entre los distintos programas, se requiere el desarrollo de una interfaz común de comunicación para el motor de juego.

Para ello, se deberán aceptar mensajes por línea de comandos según los definidos en `https://github.com/felixem/Connect6GUI` y `https://github.com/felixem/Connect6Engine`, para que sean compatibles con dicha interfaz.

Los comandos de texto son los siguientes:

1. name. Imprimir el nombre del motor de juego.
2. print. Imprimir el tablero de juego.
3. exit/quit. Terminar la partida.
4. black XXXX. Colocar la piedra(s) negra en la posición XXXX del tablero. X está en el rango A-S.
5. white XXXX. Colocar la piedra(s) blanca en la posición XXXX del tablero. X está en el rango A-S.
6. next. El motor de juego calculará la jugada para el siguiente turno.
7. move XXXX. Decirle al motor de juego que el oponente realizó el movimiento XXXX y el motor calculará el movimiento para el siguiente turno.

8. new black. Comenzar una nueva partida y otorgarle al motor de juego el jugador de negras.

9. new white. Comenzar una nueva partida y otorgarle al motor de juego el jugador de blancas.

10. depth d. Comando originalmente utilizado para indicar la profundidad máxima de búsqueda de la poda alfa-beta. En este caso, se utilizará para poner un límite de tiempo en cada turno.

11. vcf. Ignorar este comando. En el módulo de juego original se utiliza para activar el algoritmo "Victory by continuous four".

12. unvcf. Ignorar este comando. En el módulo de juego original se utiliza para desactivar el algoritmo "Victory by continuous four".

13. help. Imprimir esta ayuda.

6 Consideraciones

Se debe realizar la correspondiente experimentación para valorar el impacto que tienen sobre la fuerza de juego y la velocidad de procesamiento los parámetros de los distintos algoritmos, así como las heurísticas y optimizaciones empleadas. Toda esta experimentación debe recogerse en la memoria para que se pueda ver la evolución que ha experimentado el trabajo.

Se debe entregar una versión final del motor de juego que será el utilizado para hacer la evaluación de su rendimiento en las pruebas definidas por el profesor. Esta entrega deberá cumplir todos los requisitos de interfaz propuestos en la práctica para que otros procesos puedan interactuar con él. En caso de no cumplir estos requisitos, se considerará que el programa no funciona y supondrá el suspenso de la práctica.

Para facilitar el desarrollo colaborativo y el seguimiento de la práctica se recomienda la utilización de un sistema de control de versiones como Git. Bitbucket es una buena opción gratuita que permite la creación de repositorios privados.

Finalmente, se recuerda que la memoria es una parte muy importante de la práctica y que todas las decisiones tomadas sobre la implementación deberán recogerse y justificarse adecuadamente. En caso de entregarse un motor de juego cuya implementación no esté debidamente justificada, se considerará como si no se hubiese entregado.

7 Documentación a entregar

La documentación de la práctica es una parte muy importante en la puntuación final. El código debe estar debidamente comentado, indicando qué se hace en cada punto. La entrega final deberá constar de:

- Documentación en PDF. Se aconseja incluir las siguientes secciones: introducción, estado del arte, desarrollo, experimentación y conclusiones.
- Código Python ejecutable sin errores que cumpla completamente con los requisitos de la práctica.

8 Rúbrica de la práctica

A continuación se detalla la puntuación de los distintos apartados de la práctica:

- Memoria detallada del trabajo (40%).
- Motor de juego (35%).
- Heurística entrenada con aprendizaje automático (15%).
- Resultado del enfrentamiento contra otros oponentes (10%).

Otras mejoras adicionales que se incorporen en la práctica podrán ser valoradas positivamente según su nivel de complejidad y adecuación al problema.

9 Fecha de entrega

La fecha de entrega máxima para la práctica será el domingo 5 de noviembre a las 23:59 a través de Moodle.