Planificación automática aplicada a Lightsout

Inteligencia Artificial – Ingeniería del Software
Curso 2024/2025

Propuesta de trabajo del profesor Gabriel Chaves Benítez

1. Introducción y objetivos

El tema del trabajo se enmarca en el uso de algoritmos de búsquedas para resolver problemas de planificación. En las distintas entregas se explorarán tanto algoritmos basados en heurísticas como algoritmos estocásticos.

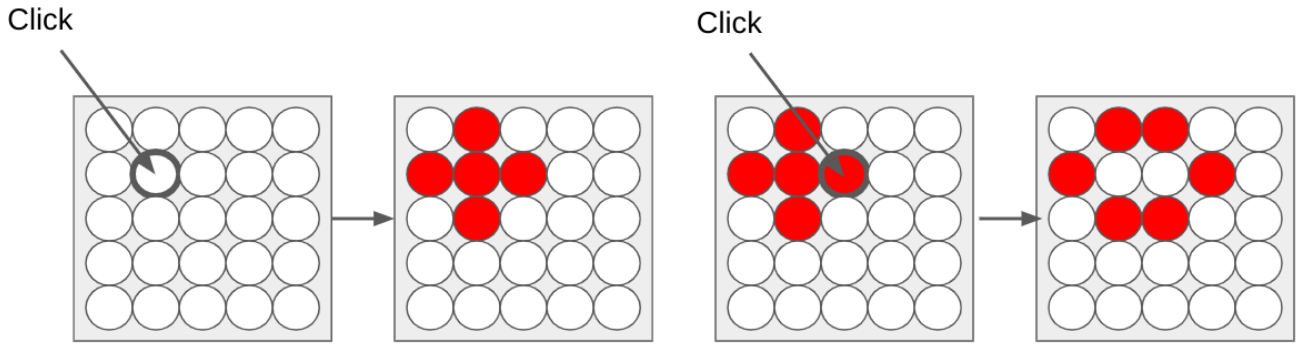
El **objetivo principal** de esta propuesta es ser capaz de diseñar un sistema que consiga representar el juego de lightsout (mediante PDDL o algún lenguaje similar como GDL) y dar con una secuencia de acciones que lleven a la solución de este. A continuación se intentará ampliar el número de tableros solucionables mediante alguno de los algoritmos que desarrollaremos posteriomente.

Para ello será necesario alcanzar los siguientes **objetivos específicos:**

1. Expresar el problema de manera correcta.
2. Diseñar un sistema capaz de usar este lenguaje y sea capaz de solucionar los tableros básicos (al menos hasta 5x5) mediante búsquedas básicas en el grafo de juego (búsqueda en profundidad, búsqueda en anchura...).
3. Aumentar el número de tableros capaces de solucionar aplicando el algoritmo correspondiente.
4. Documentar el trabajo realizado usando un formato de artículo científico.
5. Realizar una presentación (PDF, PowerPoint o similar).

2. Lightsout

El juego lightsout se enmarca dentro de los juegos de un jugador (o puzzles), este tipo de juegos son de gran utilidad en el ámbito de la IA ya que permite la evaluación del desempeño de agentes y algoritmos.



Ejemplo de movimiento en una partida

El lightsout comienza como un tablero de $N \times M$ celdas de estados binarios (o bombillas como suelen ser representadas) y en cada turno el jugador puede seleccionar una celda, esto hará que tanto ella como sus vecinos (arriba, abajo, izquierda y derecha) cambien de estado. El objetivo final del juego es, partiendo de un tablero con todas las celdas en el estado 0, cambiarlo a un tablero en el que todas las celdas estén en estado 1, idealmente en el menor número de pasos posible.

Una partida de lightsout se puede representar como un árbol en que los distintos nodos son los tableros (estados) de juego y las aristas son las acciones posibles, así, dado un nodo, los hijos serán aquellos estados accesibles a través de una acción simple.

3. Descripción del trabajo

A continuación se describe con más detalle cómo debe llevarse a cabo el trabajo. Este tendrá una parte común para todas las convocatorias y una parte específica para la convocatoria de junio y julio, en la convocatoria de octubre el alumno es libre de escoger una de las partes específicas anteriores.

3.1. Metodología

3.1.1. Parte común

En primer lugar debe expresarse el juego como un problema de planificación mediante algún lenguaje específico para ello. A continuación se implementará en Python un sistema capaz de explorar el problema, es decir, partiendo del problema anterior haga una exploración del básica del árbol de juego y sea capaz de encontrar y devolver una consecución de acciones y estados que den lugar al estado objetivo. En esta primera búsqueda será necesario ser capaz de resolver al menos el tablero básico (5×5).

3.1.2. Convocatoria de junio

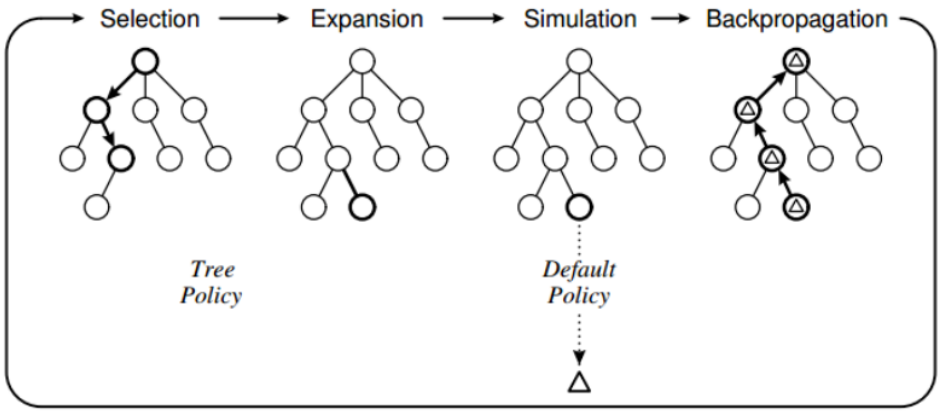
Además los alumnos que se presenten a esta convocatoria deberán crear un nuevo sistema (partiendo del anterior) que realice la búsqueda empleando el siguiente algoritmo basado en heurística.

1. Dado un estado se simulan todos los estados accesibles a través de este.
2. Si alguno de los estados accesibles es el estado objetivo se termina y se devuelve la secuencia de acciones. En otro caso a cada estado accesible se le asigna como valor el número de celdas similares al estado objetivo.
3. Se toman los n más similares y se sigue la búsqueda a partir de ellos, donde n es un número prefijado menor que la cantidad de acciones posibles.

Con este algoritmo se consigue efectuar una búsqueda en anchura pero descartando los estados "menos prometedores".

3.1.3. Convocatoria de julio

Los alumnos que se presenten a esta convocatoria deberán crear un nuevo sistema (partiendo del anterior) que realice la búsqueda mediante la siguiente versión del algoritmo de Montercarlo en su versión Tree Search:



Fases de una iteración de MCTS

El algoritmo de Monte Carlo Tree Search (MCTS) para el juego lightsout simulará varias partidas y dará una puntuación a los estados posibles en función de estas simulaciones, los pasos son:

1. **Selección del Nodo:** Partiendo del estado se selecciona un conjunto de acciones posible de manera aleatoria y realiza el procedimiento a partir del estado obtenido por ese conjunto de acciones.
2. **Expansión:** Desde el nodo seleccionado se expande el árbol de juego añadiendo un nodo **aún no visitado**.
3. **Simulación:** A partir de este nodo, se realiza una simulación del juego eligiendo acciones aleatorias. Esta simulación continua hasta alcanzar una de las siguientes condiciones de parada:

- La simulación alcanza una profundidad igual al número total de bombillas en el tablero (esta cota es una recomendación del profesor y podrá hacerse más fina si los alumnos así lo ven necesario). Si esta cota es alcanzada se le otorgará una puntuación de 0 al estado.
- Se encuentra un estado previamente visitado. Si esto ocurre, la simulación se detiene para evitar ciclos y se le otorga una puntuación de -1 al estado.
- La simulación alcanza una situación ganadora, en cuyo caso también se termina y se devuelve el camino.

4. **Retropropagación:** Tras completar una simulación se actualiza el valor de el camino recorrido de manera que el valor de un nodo es la suma del valor de sus hijos dividido entre el número de veces que fue visitado.

Este proceso se repite un número k de iteraciones para, posteriormente, efectuar una acción en función del valor de los nodos hijos y volver a iniciar el algoritmo a partir de este nuevo nodo, pero conservando la información parcial de la que ya dispone.

4. Lenguaje de programación y bibliotecas recomendadas

Para la realización del trabajo debe utilizarse el lenguaje de programación Python y se podrá hacer uso de todas las librerías vistas en la asignatura. Y para la especificación del dominio se podrá emplezar el lenguaje deseado por el alumno, aunque en caso de no ser PDDL se deberá explicar, brevemente, la sintaxis de este en la documentación del trabajo. Además, para este trabajo específicamente se recomienda:

- Unified Planning es una librería similar a Py2PDDL, pero con mayor flexibilidad en su sintaxis y mayor variedad de solver.
- Fast Downward permite resolver problemas de dominio planteado en PDDL. Recomendado para automatizar la comprobación de resultados.
- ENHSP Similar al anterior pero orientado a la solución de problemas numéricos.

5. Documentación y entrega

El trabajo deberá documentarse siguiendo un formato de artículo científico, con una **extensión mínima de 6 páginas**. En la página web de la asignatura se pueden encontrar plantillas donde se sugiere una estructura general. Estas plantillas siguen el formato de los IEEE conference proceedings, en cuyo sitio web la guía para autores ofrece información más detallada. El documento entregado deberá estar en formato PDF. Se valorará el uso del sistema LaTeX.

En el caso concreto de este trabajo, la memoria deberá al menos incluir: introducción; descripción del lenguaje de descripción de dominio elegido (si procede); descripción de los algoritmos implementados, explicando las dificultades encontradas y las decisiones de diseño adoptadas para abordarlas; experimentación y descripción de los resultados alcanzados; conclusiones; bibliografía. En ningún caso debe incluirse código en la memoria.

La entrega del trabajo consistirá de un **único fichero comprimido zip** conteniendo la memoria del trabajo, los archivos de dominios de los distintos problemas y el código implementado (ficheros py o cuadernos de Jupyter).

6. Presentación y defensa

Como parte de la evaluación del trabajo se deberá realizar una defensa del mismo, para lo que se citará a los alumnos de manera conveniente.

Al inicio de la defensa se deberá realizar una pequeña presentación (PDF, PowerPoint o similar) de diez minutos en la que participarán activamente todos los miembros del grupo que han desarrollado el trabajo. Esta presentación deberá seguir a grandes rasgos la misma estructura que la memoria del trabajo, haciendo especial mención a los resultados obtenidos y al análisis de los mismos.

En los siguientes diez minutos de la defensa el profesor procederá a realizar preguntas sobre el trabajo, que podrán ser tanto de la memoria como del código fuente.

7. Evaluación del trabajo

Para la evaluación del trabajo se tendrán en cuenta los siguientes criterios, considerando una nota total máxima de 4 puntos:

- *Memoria del trabajo* (hasta 1 punto): se valorará la claridad de las explicaciones, el razonamiento de las decisiones, el análisis y presentación de resultados y el correcto uso del lenguaje. La elaboración de la memoria debe ser original, por lo que no se evaluará el trabajo si se detecta cualquier copia del contenido.
- *Lenguaje y expresión del problema* (hasta 0.5 puntos): se valorará la adecuada parametrización del problema.
- *Búsqueda de la solución* (hasta 1.5 puntos): se valorará la implementación realizada tanto la implementación básica como la extensión del trabajo en función de la convocatoria, también se valorará positivamente, aunque no es necesaria, la mejora y ampliación de los algoritmos.
- *Código fuente* (hasta 1 punto): se valorará la claridad y buen estilo de programación, corrección y eficiencia de la implementación y calidad de los comentarios. El código debe ser original, por lo que no se evaluará el trabajo si se detecta código copiado o descargado de internet.
- *Presentación y defensa*: se valorará la claridad de la presentación y la buena explicación de los contenidos del trabajo así como, especialmente, las respuestas a las preguntas realizadas por el profesor, lo que dará lugar para cada alumno por separado a un factor multiplicativo en el intervalo [0, 1] de la nota total obtenida a partir de los apartados anteriores.

IMPORTANTE: cualquier plagio, compartición de código o uso de material que no sea original y del que no se cite convenientemente la fuente, significará automáticamente la calificación de cero en la asignatura para todos los alumnos involucrados. Por tanto, a estos alumnos no se les conserva, ni para la actual ni para futuras convocatorias, ninguna nota que hubiesen obtenido hasta el momento. Todo ello sin perjuicio de las correspondientes medidas disciplinarias que se pudieran tomar.

Uso de inteligencia artificial generativa

El uso de sistemas de inteligencia artificial generativa está permitido con las siguientes condiciones:

- Debe explicarse para qué se han utilizado esos sistemas, así como describir las entradas (*prompts*) proporcionadas a los mismos.
- En la defensa del trabajo **ambos alumnos** deben demostrar conocimiento y entendimiento de la **totalidad del trabajo**, lo que incluye las respuestas obtenidas de esos sistemas.