

Enunciado del proyecto final

CONTROL DE UN DRON DE RECONOCIMIENTO

Curso Inteligencia Artificial
Grado en Ingeniería Informática
Curso 2017-18

En esta práctica aplicaremos técnicas de búsqueda para determinar la secuencia de acciones a realizar por un dron de reconocimiento cuyo objetivo es tomar varias fotografías en distintos puntos predefinidos. El entorno en el que trabaja el dron está discretizado, como muestra la cuadrícula de la Figura 1. El dron inicia el vuelo desde su base, a la que debe retornar una vez tomadas las fotografías. Las casillas en las que aparece una imagen con un objetivo de cámara representan los puntos en los que tomar las fotografías. En el entorno puede haber distintos tipos de terreno con características diferentes.

La práctica se programará en el lenguaje de programación Python. Utilizaremos la librería de búsqueda SIMPLE-AI 0.8.1 (<https://pypi.python.org/pypi/simpleai/>). Esta librería implementa distintos algoritmos de búsqueda tanto ciega como informada, por lo que nos facilitará en gran medida la implementación de la práctica.

La práctica se divide en dos partes:

- Parte básica (7 puntos): en esta parte se resolverá un problema de reconocimiento básico en un entorno simplificado.
- Parte avanzada (3 puntos): en esta parte se añadirán características adicionales al problema, como los distintos tipos de terreno, batería de dron, etc.

1. Parte básica (7 puntos)

Consiste en resolver un problema básico de búsqueda en el que el dron debe acudir las localizaciones en las que tomar fotografías y finalmente volver a su base. Respecto a los distintos tipos de terreno, en esta parte sólo se considerará que el dron no puede sobrevolar por encima del agua. No se tendrá en cuenta ninguna información adicional sobre el entorno ni el dron (se considera batería ilimitada). El coste de la solución será el número de movimientos entre casillas que realiza el dron.



Figura 1: Ejemplo de problema: la casilla azul representa agua; el dron parte de la base (posición inicial); y las casillas con un diafragma representan los puntos en los que se deben tomar las imágenes. El recorrido ya realizado se muestra en tono más claro.

1.1. Descripción de la tarea a realizar

La tarea a realizar tiene dos etapas:

1. **Diseño e implementación:** el alumno deberá diseñar y codificar los estados, las acciones, el estado inicial y las condiciones que deben cumplir los estados meta y la heurística, para resolver el problema planteado. Además deberá elegir y ejecutar varios algoritmos de búsqueda para resolver el problema, con el objetivo de comprobar que la implementación es correcta.
2. **Experimentación y comparativa:** esta parte incluirá al menos lo siguiente:
 - Experimentación con el problema básico proporcionado aplicando distintos algoritmos de búsqueda. Se establecerá una comparativa entre los algoritmos considerando nodos expandidos, coste de la solución y memoria utilizada.
 - Se generarán al menos tres nuevos escenarios que se consideren interesantes y se repetirá el proceso del apartado anterior en estos escenarios. Se puede evaluar la diferencia entre el mismo escenario con y sin tipos de terreno que representan agua. También, se puede variar el tamaño de la cuadrícula, el número de objetivos y su localización. Se analizará cuál es el impacto de dichos parámetros en los resultados. La entrega del código no es suficiente para la evaluación de la práctica. Será necesario describir la experimentación realizada en la memoria y describir los resultados de la misma, así como los escenarios y configuraciones que se hayan considerado.

1.2. Instalación del software

Para la instalación del software se debe desempaquetar el fichero `.zip` proporcionado e instalar el paquete `pygame`¹.

Para ejecutar el software, hay que teclear: `python startGame.py`

Los ficheros que el alumno debe modificar están en la carpeta `student`. Concretamente esta carpeta contiene los ficheros:

- `config.py`: fichero que define parámetros de la configuración del juego.
- `gameProblem.py`: fichero plantilla para implementar los componentes del espacio de problemas concreto (estados, acciones, heurística,...).

1.3. Descripción del código

Para realizar la codificación se deberán implementar las funciones requeridas por el módulo de búsqueda de SIMPLE-AI, en el fichero `gameProblem.py` que proporcionamos como plantilla. El fichero `gameProblem.py` contiene la clase del mismo nombre, pero los métodos necesarios para implementar el problema están por implementar. Las secciones que debe modificar el alumno están indicadas mediante comentarios en el código.

La clase `gameProblem`, que representa el problema de búsqueda, tiene unos atributos que están descritos en el Apéndice de la práctica y permiten acceder al mapa, tipos de casilla, etc. El alumno puede hacer uso de los mismos, pero **no modificarlos**.

Para la codificación del problema, el alumno debe implementar los siguientes métodos, que son específicos para cada problema:

- `actions(s)`: devuelve una lista de acciones aplicables en cierto estado `s`.
- `result(s,a)`: devuelve el estado, resultado de aplicar la acción `a` en el estado `s`
- `is_goal(s)`: devuelve `True` si `s` es un estado final.

¹paquete `python-pygame`. En los laboratorios ya está instalado.

- *cost* (s, a, s'): devuelve un valor numérico positivo que representa el coste de ir de un estado s a otro s' aplicando cierta acción a .
- *heuristic* (s): devuelve el valor de la función heurística $h(\cdot)$ para el estado s .
- *setup*(): devuelve el estado inicial, el estado final, y algoritmo a utilizar².

2. Parte avanzada (3 puntos)

En esta parte se realizan experimentos y comparativas similares a las de la parte básica, pero considerando problemas con algunas características adicionales. Para incluir estas características el alumno deberá realizar los cambios oportunos en el diseño e implementación. Se propone considerar una o varias de las siguientes alternativas:

- **Costes del terreno.** Cada tipo de terreno tendrá un coste de atravesarlo. Este coste puede representar por ejemplo tiempo o consumo de batería (las montañas hay que sobrevolarlas, por lo que atravesarlas es más costoso que ir por terreno llano). El coste de cada tipo de terreno se puede codificar en el fichero de configuración.
- **Heurísticas.** Definir diferentes heurísticas y evaluar su efectividad.
- **Batería limitada.** Considerar que el dron tiene un nivel de carga de su batería. Cada acción consume un determinado número de unidades de carga. La carga se repone completamente en ciertas localizaciones distribuidas por el mapa. Hay que representar estas localizaciones concretas y tenerlas en cuenta en los operadores. La localización de la estación de repostaje y nivel de carga máximo se pueden especificar en el fichero de configuración añadiendo una clave adicional.

3. Entrega

La práctica se deberá realizar en grupos de dos personas y podrá ser entregada a través del enlace que se publicará en Aula Global. El nombre del fichero debe contener los 6 últimos dígitos del NIA de los alumnos (e.g., practicaIA-387633-209339.zip). Este fichero debe contener una memoria que incluya:

1. Introducción.
2. Para cada una de las partes de la práctica:
 - Descripción y explicación de la representación del problema de búsqueda diseñada.
 - Descripción de los experimentos realizados y sus resultados.
3. Conclusiones: conclusiones técnicas relacionadas con el desarrollo de esta práctica que se puedan extraer de los resultados de los experimentos que se hayan llevado a cabo.
4. Comentarios personales: opinión acerca de la práctica, dificultades, etc.

El fichero .zip al descomprimirse debe generar una carpeta con nombre *pr-ia-2017-18-student-code* que contenga todo el código de la práctica incluyendo las partes implementadas por los alumnos y un **documento .pdf** con la memoria.

²La librería de búsqueda ofrece los siguientes: *breadth_first*, *depth_first*, *limited_depth_first*, *iterative_limited_depth_first*, *uniform_cost*, *greedy*, *astar*.