Grado en Ingeniería Informática

Inteligencia Artificial

Curso 2024/2025

Universidad de Jaén

Práctica 2:
Exploración y búsqueda

1. Introducción

El objetivo de esta práctica es que el estudiante se familiarice con los agentes que implementan estrategias de exploración y búsqueda, tanto no informadas como informadas, en el ámbito de la inteligencia artificial.

Para ello se utilizará el entorno **Pacman**, el cual simula un laberinto interactivo y visual en el que los agentes se desplazan para alcanzar objetivos específicos, como la recolección de *pellets* y otros elementos característicos del juego. Este entorno introduce desafíos dinámicos, tales como la presencia de fantasmas y obstáculos que pueden alterar la trayectoria del agente, lo que obliga a adaptar y refinar las estrategias de búsqueda.

La práctica se estructura en 5 actividades:

- **Actividad 1**: Implementación de un agente explorador, cuyo principal objetivo es recorrer y mapear el mayor número posible de casillas del laberinto.
- **Actividad 2**: Análisis del comportamiento del agente explorador.
- **Actividad 3**: Implementación de un agente que utilice la estrategia de búsqueda no informada, concretamente la búsqueda en profundidad (**DFS**).
- **Actividad 4**: Implementación de un agente que aplique la estrategia de búsqueda informada **A* (BAE)**.
- **Actividad 5**: Análisis del comportamiento de los agentes buscadores (**DFS** y **BAE**) en entornos tanto mono-agente como multi-agente.

Fecha de entrega:

- **04/04/25**.

2. Pacman

2.1. Introducción

Pacman es un *programming game* en el que se debe dotar de inteligencia al agente **Pacman**, quien se desplaza por un laberinto (*ver Ilustración 1*) en busca de *pellets* y otros objetivos, mientras compite contra los fantasmas que patrullan el mapa.

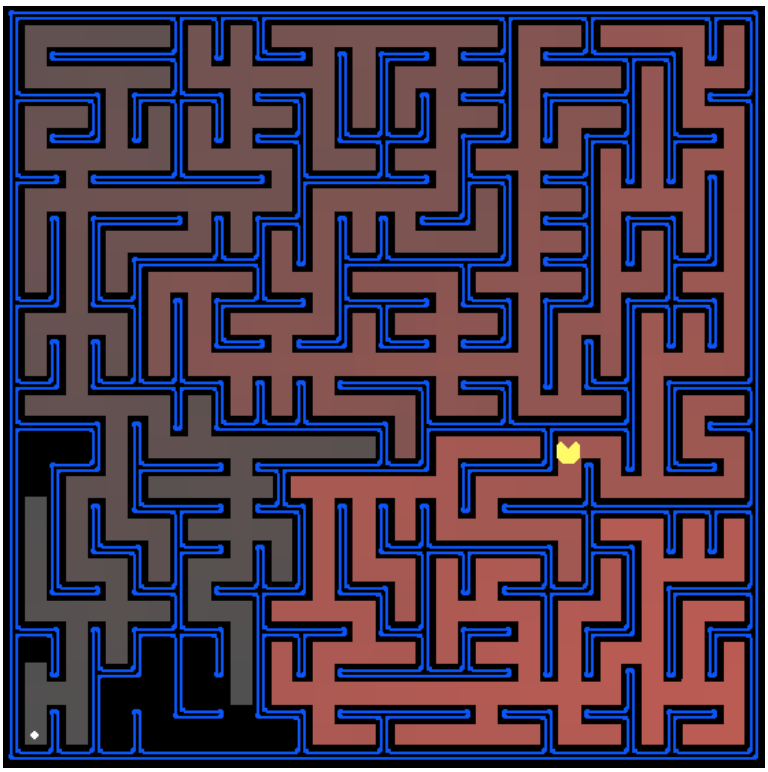

Ilustración 1. Ejemplo de ejecución de Pacman.

En esta práctica, te familiarizarás con el entorno en el que se desarrolla el juego y construirás agentes de resolución de problemas mediante estrategias de búsqueda (*agente basado en objetivos*), tomando como protagonista a **Pacman** dentro de un laberinto.

El objetivo principal del agente es encontrar y consumir los *pellets* existentes en el mapa, utilizando la información percibida para decidir el camino óptimo. Además, el juego incorpora la presencia de fantasmas y otros elementos dinámicos, de modo que la **agilidad, eficiencia y estrategia** empleadas serán esenciales para alcanzar la máxima puntuación.

En esta práctica, deberás plantear y programar diferentes estrategias que permitan explorar el laberinto y buscar los objetivos de la manera más rápida y eficaz posible. Para ello, podrás emplear las estructuras de datos y técnicas de búsqueda que consideres más adecuadas, aprovechando toda la información a la que **Pacman** tiene acceso en cada momento.

2.2. Características del entorno

El entorno de **Pacman** ha sido diseñado para facilitar la implementación y experimentación con algoritmos de búsqueda. Algunas de sus características más destacadas son:

- Requiere de **Python 3** (se recomienda Python 3.6 o superior). No se requieren dependencias externas adicionales, ya que el proyecto utiliza únicamente módulos estándar de Python.
- Permite **visualizar el laberinto**, jugar con Pacman, ver el recorrido del agente y la expansión de nodos a medida que se ejecutan los algoritmos de búsqueda. Esto ayuda a entender de forma visual cómo el agente navega por el entorno.
- El proyecto se organiza en varios archivos Python, lo que permite concentrarse en las partes relevantes para la implementación de los algoritmos (principalmente en `search.py` y `searchAgents.py`).
- Otros archivos como `pacman.py`, `game.py` y `util.py` proporcionan la lógica del juego, estructuras de datos útiles y soporte para el funcionamiento global del entorno. Aunque no se requiere su modificación, puede ser interesante visualizar estos ficheros para aprender sobre Python y el propio funcionamiento interno del juego.
- **Pacman** puede ejecutarse con distintos laberintos (*layouts*), lo que permite probar los agentes en escenarios de diferentes tamaños y niveles de complejidad. Estos niveles actualmente son preestablecidos y se encuentran disponibles en la carpeta `layouts/`.
- Las opciones de configuración se pueden ajustar fácilmente mediante **argumentos en la línea de comandos**.
- Se proporcionan agentes de ejemplo, como el `GoWestAgent`, entre otros, que permiten comprender la estructura y el funcionamiento básico de un agente.

- Cada algoritmo está implementado en una función diferente dentro del propio **Pacman** (en `search.py`), por lo que el alumno solo deberá implementar una función por cada algoritmo de búsqueda. A través de la línea de parámetros se podrá elegir la función de búsqueda a utilizar.

2.3. Instalación

Instalar y utilizar el juego es muy sencillo, ya que **no** se requieren dependencias adicionales. Para ello, se deben seguir los siguientes pasos:

1. **Descarga** el archivo comprimido de PLATEA que contiene todo el código fuente y los archivos de soporte del proyecto.
2. **Extrae** el contenido del archivo en una carpeta de tu elección.

Ejecución del juego:

- Abre una **terminal** y navega hasta el directorio donde se extrajo el proyecto.
- Para iniciar un juego de Pacman con la configuración por defecto, ejecuta:

  ```
  python pacman.py
  ```

- Para iniciar una partida utilizando un *layout* específico y un agente en concreto (por ejemplo, el `GoWestAgent`), puedes usar:

  ```
  python pacman.py --layout testMaze --pacman GoWestAgent
  ```

Un parámetro importante para poder utilizar algunos laberintos adecuadamente es la opción `-k` (o `--numghosts`), que indica el número de fantasmas que aparecen en el mapa. En nuestro caso, **NO debemos tener fantasmas**, por lo que debemos desactivarlos incluyendo el argumento `-k 0` en nuestras ejecuciones.

También, si no vemos el mapa de manera adecuada, se puede **ampliar/reducir** su tamaño incluyendo un factor de *zoom* en la ejecución. Esto se hace a través de la opción `-z`. Se recomienda que en los mapas más grandes, se use la opción:

```
python pacman.py -z 0.5
```

Esto reducirá su tamaño a la mitad, permitiendo visualizarlo correctamente en pantalla.

Si necesitas ayuda sobre las opciones disponibles, ejecuta:

```
python pacman.py -h
```

Durante la práctica, se deberá trabajar principalmente en los archivos `search.py` y `searchAgents.py`. **Asegúrate de no modificar el resto de archivos**, ya que únicamente se deben enviar estos dos ficheros en la entrega final.

2.4. Creación de un nuevo agente de búsqueda

Para crear tu nuevo agente de búsqueda en **Pacman**, deberás trabajar en dos ficheros principales: `search.py` y `searchAgents.py`.

- En `search.py` se definen las funciones genéricas de búsqueda (por ejemplo, **DFS, BFS, UCS y A***).
- En `searchAgents.py` se implementan los agentes que utilizan dichas funciones para planificar rutas en el laberinto.

Sigue estos pasos para crear tu propio agente:

1. Definir tu función de búsqueda (en `search.py`):
    1. **Abre** el fichero `search.py` y localiza la plantilla de la función de búsqueda que deseas implementar o modificar.
    2. Cada función de búsqueda (por ejemplo, `depthFirstSearch`, `breadthFirstSearch`, etc.) debe adherirse al siguiente formato:
        - **Entrada**: Un objeto que implemente la interfaz `SearchProblem`, con los siguientes métodos:
            - `getStartState()`
            - `isGoalState(state)`
            - `getSuccessors(state)`
            - `getCostOfActions(actions)`
        - **Salida**: Una **lista de acciones** (*direcciones*) que llevan al agente desde el estado inicial hasta alcanzar el objetivo.
    3. **Utiliza** las estructuras de datos proporcionadas en `util.py` (por ejemplo, `Stack`, `Queue` o `PriorityQueue`) para evitar ciclos mediante la implementación de una **búsqueda en grafo**.

2. Crear el agente en `searchAgents.py`:
    1. **Abre** el fichero `searchAgents.py` y crea una **nueva clase** para tu agente, por ejemplo, `FooSearchAgent`, que herede de la clase base (**usualmente `SearchAgent`**).
    2. En el **constructor** de tu clase, realiza lo siguiente:
        - **Inicialización**: Llama al constructor de la superclase, configurando el nombre de tu agente (por ejemplo, `"FooSearchAgent"`). Este nombre te ayudará a identificarlo en el juego.
        - **Asignación de la función de búsqueda**: Especifica la función de búsqueda que utilizará el agente, la cual debe haber sido definida o modificada en `search.py`.
3. Registro y prueba de tu nuevo agente:
    – Asegúrate de que tu agente esté **correctamente definido y registrado** para que el entorno **Pacman** lo reconozca.
    – Para **probar tu agente**, ejecuta en la terminal el siguiente comando, utilizando un *layout* adecuado (por ejemplo, `testMaze`):

```
python pacman.py --layout testMaze --pacman FooSearchAgent
```

Movimientos legales de Pacman

- Es importante que tu agente retorne **únicamente movimientos legales**. En caso contrario, el juego te expulsará. Estos movimientos se definen en el fichero `game.py`, en la clase `Directions`. Los movimientos disponibles son:
    – **NORTH**: Representa el movimiento hacia el norte.
    – **SOUTH**: Representa el movimiento hacia el sur.
    – **EAST**: Representa el movimiento hacia el este.
    – **WEST**: Representa el movimiento hacia el oeste.
    – **STOP**: Representa la acción de no moverse.
- Además, la clase `Directions` define **diccionarios auxiliares** para obtener las direcciones relativas:
    – **LEFT**: Mapea cada dirección a la dirección que se obtiene al girar a la izquierda.

  - **RIGHT**: Mapea cada dirección a la dirección que se obtiene al girar a la derecha.
  - **REVERSE**: Indica la dirección opuesta a la actual.
- Asegúrate de que los **algoritmos de búsqueda** y las funciones implementadas en `search.py` generen **únicamente estas direcciones** al planificar el movimiento de Pacman.

Con estos pasos, habrás **creado e integrado un nuevo agente de búsqueda** en el entorno **Pacman**, aprovechando la funcionalidad definida en `search.py` para planificar rutas mediante algoritmos de búsqueda y asegurándote de que los movimientos que realice sean **siempre legales**.

2.5. Información adicional sobre los ficheros

Durante la partida, **Pacman** mostrará estadísticas que te permitirán **evaluar la eficiencia** de tu estrategia de búsqueda.

Además, a continuación, se da una **descripción general** de los archivos más importantes de **Pacman**. Se recuerda que **no se deben editar estos ficheros**, pero es recomendable revisarlos para obtener una mejor visión general del sistema.

Descripción de los archivos principales

1. **`game.py`**:
   Define el **motor del juego** y contiene las clases y estructuras principales del juego, como:
   - `Agent`
   - `Directions`
   - `Configuration`
   - `AgentState`
   - `Grid`
   - `Game`
2. **`pacman.py`**:
   Contiene la **lógica principal del juego Pac-Man**, incluyendo:
   - `GameState`
   - Las reglas del juego

3. **`ghostAgents.py`**:
   Implementa el comportamiento de los **fantasmas** en el juego, permitiéndoles:
   - Moverse aleatoriamente
   - Perseguir a **Pac-Man**
4. **`pacmanAgents.py`**:
   Define diferentes estrategias para **Pac-Man**, como:
   - Girar siempre a la izquierda
   - Seguir una estrategia *greedy*

3. Actividades relativas al entorno y la exploración

3.1. Actividad 1. Implementación de un agente explorador

En esta actividad, los estudiantes deberán implementar un **agente inteligente** en el entorno de **Pacman**, con el objetivo principal de **explorar el mayor espacio posible del laberinto** mientras se aproxima a su objetivo, que en este caso es **recoger pellets** (u otro objetivo similar definido en el *layout*).

> **Nota:**
> El propósito de esta actividad **no es necesariamente recoger la comida antes que los demás agentes**, sino **evaluar y optimizar la capacidad exploratoria** del agente en el laberinto.

Para implementar este agente, los estudiantes deberán:

1. **Crear una nueva subclase** dentro del fichero correspondiente (`searchAgents.py`).
2. **Definir una nueva función llamada `exploracion` en `search.py` que recoja la lógica de la exploración del agente.**

Consideraciones para el diseño del agente

- El agente **debe ajustarse a la variabilidad del entorno**, sin depender de información externa o de parámetros predefinidos del laberinto.
- **Debe basar su comportamiento únicamente en el estado actual** recibido y su **memoria interna**.
- Debe gestionar internamente:
  - **El registro de las celdas visitadas**.
  - **Los métodos y atributos auxiliares** necesarios para maximizar su capacidad de exploración.

El código del proyecto incluye **algunos agentes de ejemplo** en `pacmanAgents.py`, los cuales pueden servir de **referencia** para entender la

estructura y el comportamiento esperado de un agente explorador. Se recomienda:

- **Estudiar dichos ejemplos**.
- **Utilizarlos como punto de partida** para diseñar una estrategia que **priorice la exploración del laberinto sin descuidar la aproximación hacia el objetivo final**.

3.2. Actividad 2. Análisis del comportamiento del agente explorador

Una vez que hayas implementado tu primer **agente explorador**, **ejecuta el juego en distintos laberintos** para evaluar su desempeño.

Se recomienda realizar pruebas en, al menos, los siguientes escenarios:

- `smallMaze`
- `mediumMaze`
- `bigMaze`
- `bigSafeSearch`
- `originalClassic`
- `powerClassic`
- `trickySearch`
- `greedySearch`

⚠️ **NOTA IMPORTANTE:**
Recuerda incluir el parámetro `-k 0` en cada ejecución del mapa para que **no aparezcan fantasmas** que entorpezcan tu búsqueda.

Análisis de desempeño

Al finalizar cada partida, se deben **mostrar una serie de estadísticas**, entre las que se incluyen:

- **Cantidad de pasos realizados**.
- **Número de celdas únicas visitadas** (este dato debe ser gestionado e implementado por el estudiante dentro del agente).
- **Ratio de repetición**: que es el cociente entre los pasos realizados y las celdas únicas visitadas.

Con estos datos, deberás **construir una tabla de análisis**, como la que aparece a continuación, con las siguientes columnas:

Laberinto	Coste acumulado	Total de pasos	Número de casillas exploradas	Ratio de repetición
`smallMaze`	X	X	X	X
`mediumMaze`	X	X	X	X
`bigMaze`	X	X	X	X

4. Estrategias de búsqueda

Ya hemos realizado un **agente con capacidad de exploración alta**, que nos ha permitido **explorar el entorno** y familiarizarnos con el juego.

A continuación, nuestro objetivo será desarrollar un **agente capaz de elaborar varias estrategias de búsqueda**. En particular, en esta sección se diseñarán dos algoritmos:

1. **Un algoritmo de búsqueda no informada** (DFS).
2. **Un algoritmo de búsqueda informada** (A*).

4.1. Actividad 3. Implementación de un agente con estrategia de búsqueda en profundidad (DFS)

En esta actividad, los estudiantes implementarán un agente que utilice el algoritmo de **Búsqueda en Profundidad** (*Depth-First Search*, *DFS*), conforme a los conceptos estudiados en la parte teórica de la asignatura.

Instrucciones de implementación

- La **clase del agente** deberá mantener el mismo nombre que la del **explorador**, pero con el sufijo `"dfs"`.
- La implementación podrá hacer uso de **estructuras de datos necesarias** para gestionar la exploración del entorno.
- Se podrá utilizar **toda la información que el juego proporciona** al agente en cada momento.

Nota importante: No está permitido el uso de funciones heurísticas, ya que **DFS es un algoritmo de búsqueda no informada**. Se puede utilizar la **posición del objetivo** únicamente para **verificar la condición de objetivo alcanzado**, pero **no** para estimar distancias ni optimizar la exploración.

4.2. Actividad 4. Implementación de un agente con la estrategia de búsqueda A* (BAE)

En esta actividad, los estudiantes implementarán el algoritmo de **A*** (*BAE*), estudiado en la parte teórica de la asignatura.

Instrucciones de implementación

- La **clase del agente** se llamará igual que la del **explorador**, pero añadiendo el sufijo `"bae"`.
- Se podrán utilizar **las estructuras de datos necesarias** para gestionar la búsqueda.
- Se podrá utilizar **toda la información que el juego facilita** al agente en cada momento.
- Para hacer uso de la **función heurística**, se recomienda **crear una función aparte** para este propósito.

Nota importante: En esta actividad, se usa un **algoritmo de búsqueda informada**. **Es obligatorio definir una función heurística** para estimar la **distancia de una posición cualquiera a la posición del objetivo**.

4.3. Actividad 5. Análisis del comportamiento del agente buscador

Se ejecutará un análisis con la **misma configuración** que se utilizó para el **agente explorador**, pero ahora aplicado a los **agentes buscadores** (`DFS` y `BAE`).

El análisis debe incluir, al menos, los siguientes aspectos:

1. Comportamiento en entorno mono-agente - Evaluar el desempeño de los agentes `DFS` y `BAE` cuando **no hay fantasmas en el laberinto**. - Comparar métricas como: - **Tiempo total de ejecución**. - **Cantidad de pasos realizados**. - **Número de nodos explorados**. - **Eficiencia en la búsqueda del objetivo**.

2. Comportamiento en entorno multi-agente - Evaluar el desempeño de los agentes `DFS` y `BAE` cuando **hay fantasmas en el laberinto**. - Analizar cómo afecta la presencia de enemigos a la: - **Estrategia de búsqueda**. - **Eficiencia en la toma de decisiones**. - **Capacidad de alcanzar el objetivo sin ser atrapado**.

3. Dificultades y posibles soluciones - Identificar **las dificultades** que enfrentan los agentes `DFS` y `BAE` en un **entorno multi-agente**. - Proponer **soluciones o ajustes** que puedan mejorar su desempeño, tales como: - Incorporación de una estrategia de evasión de fantasmas. - Adaptación de la búsqueda para evitar zonas peligrosas. - Uso de heurísticas mejoradas en `BAE` para optimizar la planificación de rutas.

5. Consideraciones importantes

- Es necesario **generalizar el comportamiento del agente** para que su funcionamiento sea **adecuado en todo tipo de escenarios**.
- En la **implementación del agente con la estrategia de búsqueda A*** (`BAE`), se debe garantizar que la búsqueda realizada obtenga una **solución completa y óptima**.

En la práctica, se valorará:

- **Explicación o descripción de las estrategias** seguidas por el **agente explorador** en su exploración.
- **Idea o lógica** detrás de las estrategias utilizadas.
- **Efectividad** de las estrategias aplicadas.
- **Calidad del código**, incluyendo:
  - **Limpieza del código entregado**.
  - **Documentación interna** del código.
  - **Documentación externa** de cada agente en el informe.
  - **Uso correcto de convenciones de Python** (`PEP8`).
- **Evitar problemas en la ejecución**, tales como:
  - Bucles infinitos.
  - Movimientos repetitivos innecesarios.
  - Descalificaciones por parte del juego.

6. Entrega y evaluación

- La puntuación máxima de la **Práctica 2** será de **4 puntos**, distribuidos de la siguiente forma:
  - **Actividad 1**: 0,75 puntos
  - **Actividad 2**: 0,25 puntos
  - **Actividad 3**: 1 punto
  - **Actividad 4**: 1,25 puntos
  - **Actividad 5**: 0,75 puntos
- La entrega de las actividades se realizará a través de la **tarea asociada en PLATEA** con fecha límite el **04/04/25**.
- La **defensa de ambas partes** de la práctica se realizará en la **sesión de prácticas siguiente a la entrega**. En ella, se llevará a cabo una **prueba escrita a ordenador**, donde se deberá demostrar el conocimiento sobre el trabajo realizado.

Formato de entrega

Cada trabajo práctico debe estar **comprimido en un archivo `.zip`**, incluyendo **documentación y código fuente**.

- Dentro del código fuente, el equipo deberá **incluir únicamente** los ficheros:
  - `search.py`
  - `searchAgents.py`

  Todos los métodos y agentes implementados deberán estar contenidos en estos archivos.
- El **nombre del archivo `.zip` y del documento PDF** deberá seguir el siguiente formato:
  `M25N_Ape11_Ape12_Ape21_Ape22.[zip|pdf]`
  Donde:
  - `N` es el número del grupo al que perteneces:
    - **Grupo 1**: 8:30 - Luis Gonzaga
    - **Grupo 2**: 10:30
    - **Grupo 3**: 12:30
    - **Grupo 4**: 15:30
    - **Grupo 5**: 17:30
    - **Grupo 6**: 8:30 - José María Serrano

        - **Grupo 7**: 19:30
    - `Ape11` y `Ape12` son el **primer y segundo apellido** del **primer integrante** del equipo.
    - `Ape21` y `Ape22` son el **primer y segundo apellido** del **segundo integrante** del equipo.
- Además del código, se entregará un **documento en formato PDF** que describa el desarrollo de la práctica:
    - **Explicación de la estrategia seguida en cada uno de los métodos implementados** en lenguaje natural.
    - **No debe incluir código fuente**, pero sí se pueden usar **pseudocódigo o esquemas**.
    - **Rellenar las tablas y razonar brevemente los resultados**.
    - **Extensión recomendada**: 4-6 páginas.

Cada documento debe comenzar con una **portada** que incluya, al menos, la siguiente información:

- **Nombre de los autores**.
- **Curso académico**.
- **Grupo de prácticas**.
- **Nombre del profesor de prácticas**.
- **Índice**.

El incumplimiento de las normas anteriores repercutirá negativamente en la calificación de la práctica.