Práctica: *Búsqueda*
Agentes Inteligentes
Grado en Robótica. Curso 2023-2024.
Departamento de Electrónica y Computación

1. Objetivo

El objetivo de esta práctica es que el alumno aprenda a implentar algoritmos de búsqueda y aplicarlos a problemas robóticos.

2. Enunciado del problema

La empresa *Having a Clean House is Important S.L.* os ha contactado a tu compañero y a ti para controlar sus robots limpiadores de forma que se muevan de manera efectiva por las habitaciones de la casa mientras aspiran la suciedad. Para resolver este problema, se dispone de los robots que se muestran en la Figura 1. Como se puede apreciar, se trata de robots circulares sencillos con dos ruedas que permiten maximizar la superficie cubierta por los robots durante las tareas de limpieza.

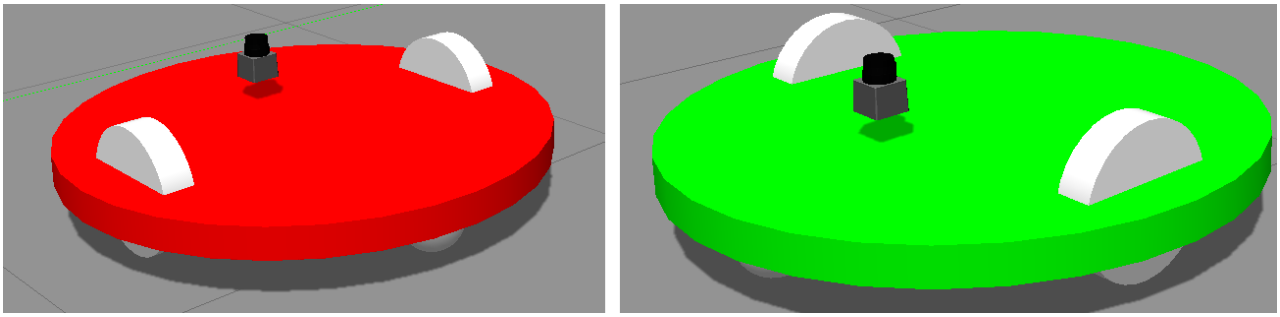

Figura 1: Robot limpiadores.

Uno o varios de estos robots se moverán por las habitaciones de la casa recogiendo la suciedad. Una de estas habitaciones se muestra de manera esquemática en la Figura 2. Esta habitación se ha dividido en una cuadrícula o *grid*, donde cada celda representa una posible localización donde podría encontrarse un robot. A su vez, existen diferentes tipos de celdas dentro de este *grid*. Las celdas amarillas representan celdas sucias por las que el robot debe pasar recogiendo la suciedad; la celda del mismo color que el robot representa un punto de carga donde puede recargar su batería antes de continuar con el proceso de limpieza; las celdas blancas representan puntos donde el robot puede vaciar su depósito de suciedad; por último, también existen celdas con obstáculos no transitables. Cada robot, por tanto, dispone de una batería y de un depósito donde almacena la suciedad que recoge. El valor inicial de la batería de cada robot es configurable, y se reduce en un determinado valor también configurable cada vez que el robot realiza una acción. El valor inicial del depósito de suciedad de cada robot también es configurable, y se incrementa en un determinado valor también configurable cada vez que el robot llega a una celda sucia. Si durante el proceso de limpieza el robot no cuenta con la suficiente batería para completar sus operaciones, debe acudir a un punto de carga para así poder continuar. De manera similar, si el depósito de suciedad de un robot está lleno, deberá acudir a un punto de vaciado para poder continuar limpiando celdas. En cuanto a las operaciones de carga de batería y vaciado del depósito, se asume que cuando el robot llega a una celda de carga, su batería se llena completamente, y cuando llega a una celda de vaciado su

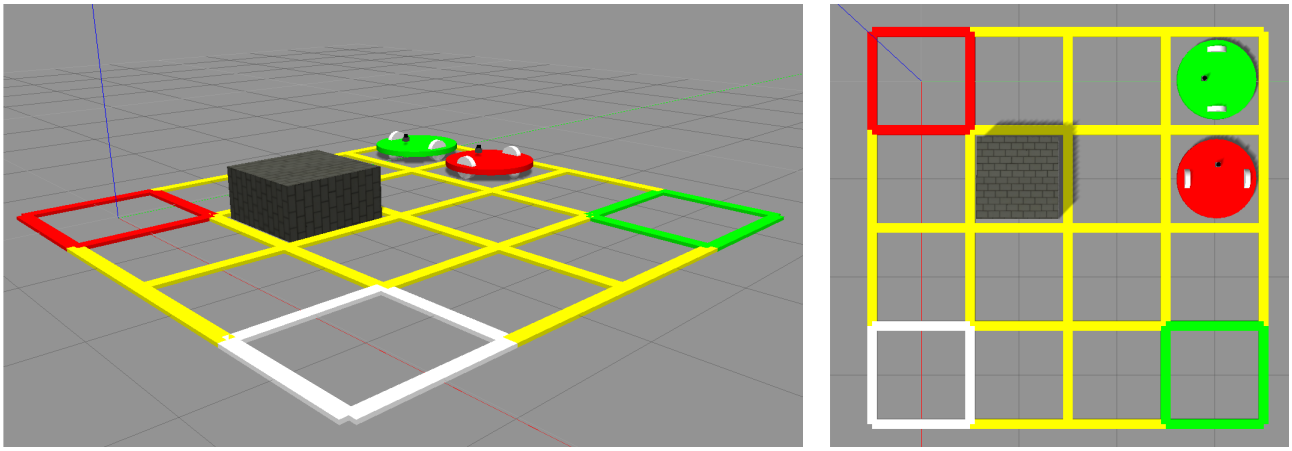

Figura 2: Representación de una habitación.

depósito de suciedad se vacía por completo. Asimismo, se asume que cuando un robot llega a una celda sucia, la limpia, y no es necesario volver a limpiarla.

Con respecto a la navegación dentro de los almacenes, cada robot solo puede realizar tres acciones diferentes: moverse una celda hacia delante, y girar izquierda o derecha. En este video se puede observar como los robots llevan a cabo las tareas de limpieza en la habitación propuesta realizando movimientos aleatorios. **Es importante apreciar que los robots realizan sus acciones en paralelo, es decir, en cada paso se ordena a cada uno de los robots que realicen una acción**. Además, no es necesario representar explícitamente las operaciones de carga de batería, vaciado de depósito y limpieza de celda, ya que como se mencionaba anteriormente, se asume que estas operaciones se llevan a cabo implícitamente cuando el robot llega a la celda correspondiente. También como se comentaba anteriormente, los costes asociados a las acciones de movimiento o giro son configurables por cada robot, pero se pueden asumir los siguientes:

- **Mover**: coste de 1.
- **Giro**: coste de 0.5.

Estos costes se pueden entender como la cantidad de unidades en la que se decrementa la batería del robot cada vez que hace la acción correspondiente. **El objetivo por tanto, es minimizar el coste total empleado por los robots para limpiar todas las celdas sucias de una habitación.** Este objetivo también se puede entender como que cada celda amarilla debe ser visitada al menos una vez por un robot que no tenga el depósito lleno, de forma que pueda recoger la suciedad de la celda correspondiente. **Aunque la posición inicial de los robots en la habitación puede ser cualquiera, su posición final debe ser sus puntos de carga correspondientes.**

Teniendo todo esto en cuenta se pide:

1. Modelar el problema de planificación de limpieza de habitaciones como un problema de búsqueda. Para ello, el alumno deberá buscar una representación para el espacio de estados, el estado inicial y final, y definir los operadores (con sus precondiciones y efectos) que permita llevar a cabo el proceso de búsqueda (Figura 3).

2. Implementar un algoritmo de búsqueda (e.g., amplitud, profundidad, profundidad iterativa, A*, IDA*)[1] que permita resolver el problema. En el caso de que se decida implementar A* o IDA*, implementar al menos una función heurística informada y admisible que estime el coste restante, de modo que sirva de guía para el algoritmo. Se recomienda comenzar con configuraciones de habitaciones sencillas, con dos robots, sin obstáculos, y habitaciones pequeñas que permitan probar y depurar el código. Después se recomienda ir incrementando la dificultad de las habitaciones.

[1] Si se decide implementar un algoritmo de búsqueda no informada, se puede asumir que el coste de todas las operaciones del robot descritas anteriormente (desplazamiento y giros) son unitarios.

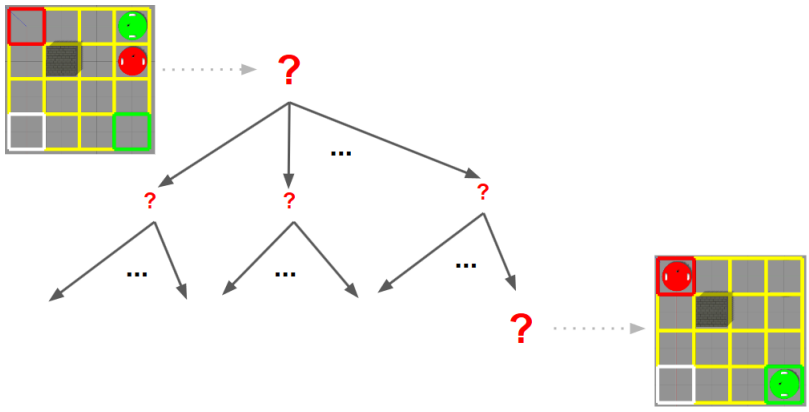

Figura 3: Representación del proceso de búsqueda.

3. La implementación desarrollada se deberá ejecutar desde una consola o terminal con el siguiente comando:

   ```
   python practica2.py
   ```

4. El programa debe generar dos ficheros de salida: `plan.txt` y `statistics.txt`. Ambos se deben generar en el mismo directorio donde se encuentre el programa `practica2.py`. Los ficheros son los siguientes:

   - Solución del problema (`plan.txt`). Debe contener las operaciones que realizan los robots para llevar a cabo la limpieza de las habitaciones. Un ejemplo puede ser:

     ```
     move robot1; move robot2
     rotateLeft robot1; move robot2
     none robot1; rotateRight robot2
     ...
     move robot1; rotateLeft robot2
     ```

     donde cada línea representa un instante de tiempo diferente.

   - Fichero de estadísticas (`statistics.txt`). Este fichero debe contener información relativa al proceso de búsqueda, como el tiempo total, coste total, longitud del plan, nodos expandidos, etc. Por ejemplo,

     ```
     Tiempo total: 145
     Coste total: 54
     Longitud del plan: 27
     Nodos expandidos: 132
     ```

   donde `Tiempo total` se refiere al tiempo total desde que comienza la ejecución del algoritmo de búsqueda hasta que encuentra una solución, `Coste total` es el coste acumulado de los costes individuales de las acciones que componen el plan, `Longitud del plan` es el número de acciones en el plan, y `Nodos expandidos` se refiere al número total de nodos que han sido expandidos.

5. Proponer casos de prueba con diversas configuraciones de habitaciones y resolverlos con la implementación desarrollada. Estos casos se deben generar razonablemente dependiendo de la eficiencia alcanzada

en la implementación, y deberían crecer en dificultad hasta encontrar uno que no sea capaz de solucionar en un tiempo razonable.

6. En el caso de que se implemente A* o IDA*, realizar un estudio comparativo utilizando la(s) heurística(s) implementada(s) (número de nodos expandidos, tiempo de cómputo, etc.). En particular, se puede realizar un estudio entre la(s) heurística(s) diseñada(s), y una heurística que es 0 en todos los casos (*Dijkstra*).

3. Directrices para la Memoria

La memoria debe entregarse en formato .pdf y tener un máximo de 15 hojas en total, incluyendo la portada, contraportada e índice. Al menos, ha de contener:

1. Breve introducción explicando los contenidos del documento.
2. Descripción del modelado del problema como un problema de búsqueda, describiendo la representación del espacio de estados, operadores (con sus precondiciones y efectos), y, en el caso de A* o IDA*, los costes de la aplicación de los operadores, y función heurística diseñada.
3. Análisis de los resultados. Estudio del rendimiento de la implementación desarrollada en una variedad significativa de configuraciones de almacenes que crezcan en complejidad para determinar cómo escala la solución propuesta. En el caso de A*/IDA*, además, análisis de la(s) heurística(s) propuestas y comparación con la heurística igual a 0 en todos los casos.
4. Conclusiones acerca de la práctica.

La memoria **no debe incluir código fuente** en ningún caso.

4. Evaluación

La evaluación de la práctica se realizará sobre 10 puntos. La distribución de puntos es la siguiente:

1. Modelización del problema de búsqueda (4 puntos)
2. Implementación del algoritmo de búsqueda. Se valorará muy positivamente que se decida implementar A* o IDA* (4 puntos)
3. Análisis de resultados (2 puntos)

5. Entrega

Se tiene de plazo para entregar la práctica **hasta el 12 de Noviembre a las 23:55**. Sólo un miembro de cada pareja de estudiantes debe subir a la sección de prácticas de Campus Virutal un único fichero `.zip`. Es importante cumplir las siguientes instrucciones para la entrega.

1. El fichero debe nombrarse `p2-Apellido1-Apellido2.zip`, donde `Apellido1` y `Apellido2` se corresponden con el primer apellido de los integrantes del grupo de trabajo. La descompresión de este fichero debe contener:

   a) La memoria en formato .pdf con nombre `memoria.pdf`

   b) Carpeta "`vacuum_cleaner`" que se corresponde con el paquete que se utilizará para desarrollar la solución. Al inicio de la práctica se proporciona una primera versión de este paquete, que se utilizará como base para construir los desarrollos requeridos. El paquete final que se entregue debe contener además de todo el código desarrollado por los alumnos, todos las habitaciones propuestas para llevar a cabo el análisis de los resultados.

Importante: no seguir las normas de entrega puede suponer una pérdida de hasta 1 punto en la calificación final de la práctica.

6. Indicaciones

En esta sección se describen algunas indicaciones para realizar la práctica:

1. Al inicio de esta práctica se proporciona un paquete "`vacuum_cleaner`" con la descripción del escenario utilizado como ejemplo en este enunciado, así como un fichero "`launch`" que permite la carga del mundo y los robots. Este paquete se puede encontrar en la carpeta "`Recursos`" de Campus Virtual con el nombre "`vacuum_cleaner.tar.gz`".

2. Aunque no es obligatorio, es recomendable que los alumnos trabajen con una máquina virtual que hemos habilitado para el desarrollo de las prácticas. Para usarla convenientemente:

   - Importar la máquina virtual con Virtual Box. **En las opciones de importación seleccionar un disco externo o memoria USB con suficiente capacidad para no ocupar disco local**. Procediendo de esta forma el alumno podrá llevar consigo la máquina virtual con los cambios realizados y no tendrá que preocuparse de **borrar la máquina virtual del disco local cada vez que acabe la sesión de prácticas**.
   - Independientemente de lo anterior, se recomienda **hacer copias de seguridad periódicas del directorio que contiene el paquete de trabajo que se está desarrollando** ya sea en `github` o en una memoria USB
   - Una vez realizada la importación, cargar la máquina virtual. La clave para acceder a la sesión es `robotica`.
   - Abrir el fichero `.bashrc` y comentar o eliminar la línea que se encuentra al final del mismo:
     `\rm ~/catkin_ws/src/*`
   - Es conveniente regenerar el workspace de `catkin` para evitar futuros problemas. Para ello, abrir un terminal, e introducir la siguiente secuencia de comandos:
     ```
     rm -rf catkin_ws
     mkdir -p catkin_ws/src
     cd ~/catkin_ws
     catkin_make
     ```
   - Descomprimir el fichero "`vacuum_cleaner.tar.gz`" en `~\catkin_ws\src`:
     `tar -xvzf vacuum_cleaner.tar.gz`
   - Ahora hay que compilar el paquete "`vacuum_cleaner`", y hacerlo visible a ROS. Para ello, ejecutar la siguiente secuencia de comandos:
     ```
     cd ~\catkin_ws
     catkin_make
     source devel\setup.bash
     ```
   - Comprobar que el paquete "`vacuum_cleaner`" es visible para ROS:
     `rospack find vacuum_cleaner`
   - y si lo es ejecutar:
     `roslaunch vacuum_cleaner main.launch`

   Importante: El paquete proporcionado incluye todas las funciones necesarias para llevar a cabo la navegación del robot: moverse hacia adelante en el grid y girar tanto a derecha como a izquierda 90º. Para construir el proceso de búsqueda requerido, el alumno puede apoyarse en todas estas funciones ya definidas. En el fichero `practica2.py` se muestra un ejemplo donde los robots ejecutan acciones aleatorias. **Estudiar detenidamente el código de este fichero, puesto que representa el punto de partida del proyecto**. En `practica2.py` también se incluye la función `executeSearch`. Será tarea del alumno implementar su código de búsqueda en esta función, **para lo cual puede generar las clases, funciones, y ficheros adicionales que considere necesarios.**