*Alberto Quesada Aranda*

# *Práctica 1: Agentes Reactivos*

1. Análisis del problema (entorno, características del agente, etc.)

Nos encontramos con un mapa cuya geografía y posición inicial son desconocidas; a priori lo único que sabemos es que su tamaño será de 10x10 y esta cerrado. Cada casilla del mapa puede tener o una pared, o una casilla vacía (éstas se van ensuciando con el tiempo).

Nuestro agente puede moverse arriba, abajo, a la izquierda y a la derecha; además si detecta que la casilla en la que se encuentra está sucia, la limpia. Tiene también un sensor de choque que se activa cuando encuentra un obstáculo en la casilla a la que se quería mover.

No podemos detectar si una casilla esta limpia, sucia, o tiene un obstáculo hasta que nos movemos a ella. Además aunque ya hayamos limpiado una casilla puede volver a ensuciarse con el tiempo.

2. Descripción de la solución planteada

La solución más eficiente que se me ocurre empieza por reconocer el terreno y guardar la información que vaya obteniendo a través de los sensores en memoria. Dos de las restricciones del problema son que los mapas son de un tamaño fijo de 10x10 y que no se en que posición empieza el agente; por tanto, para poder guardar toda la información del mapa explorado creo una matriz (*mapa*) de tamaño 20x20, y supongo que mi posición de inicio es la *mapa[10][10]*. Así, empiece en la posición que empiece, tendré espacio suficiente en la memoria que he reservado para guardar los datos. En un principio toda la matriz está inicializada a -1.

Para saber en todo momento la posición del agente uso las variables *posx* y *posy*. Como no podemos saber qué casillas están sucias antes de ir a ellas, para decidir a que casilla nos tenemos que mover usamos un orden de prioridad, de forma que los números más bajos indican prioridad más alta (con esto consigo que en cada momento vaya a la casilla que más tiempo lleve sin ir de todas las adyacentes).

Para dar una prioridad relativa a cada celda libre uso la variable *contador*; la cual se inicializa a 0 y aumenta de uno en uno cuando el agente se mueve a una celda libre. Por tanto, el robot se moverá a la casilla adyacente a la que se encuentre en ese momento cuyo número de prioridad sea más bajo.

Al principio el robot se va moviendo aleatoriamente por el mapa y realizando las siguientes acciones:

- Si se activa la señal de choque, pone en la posición correspondiente de la matriz (*mapa*) el valor '*wall*' (pared). De esta forma, ya no volverá a chocar otra vez con la misma pared.
- Si no se activa la señal de choque, es que la casilla esta libre y por tanto coloca en la posición de la matriz que corresponda el número de prioridad indicado por la variable contador.

A medida que va "conociendo" el *mapa*, deja de moverse de forma aleatoria y para decidir hacia dónde moverse en cada momento comprueba la prioridad de todas las casillas adyacentes y se mueve a la que tenga el número más bajo.

3. Resultados obtenidos por la solución aportada en los distintos mapas.

| | *agent.map* | *mapa1.map* | *mapa2.map* | *mapa3.map* |
|---|---|---|---|---|
| *DirtyDegree* | 83153 | 132657 | 114515 | 168172 |
| *ConsumedEnergy* | 2928 | 3031 | 2989 | 3049 |

* Todos los resultados son los obtenidos al realizar 2000 acciones en cada ejecución.

4. Código fuente de los ficheros agent.cpp y agent.h

Los códigos de estos ficheros se encuentran en la carpeta contenida en el .zip -> "Códigos".