INTELIGENCIA ARTIFICIAL

E.T.S. de Ingenierías Informática y de Telecomunicaciones
Universidad de Granada

Práctica 2
Agentes reactivos
(El robot trufero)

Gregorio Vidoy Fajado
D.N.I 74733073 R
2º A2
17-04-2015

1.- Descripción de la solución planteada.

1.1 Método de resolución empleado.

Para la resolución del problema, e implementado memoria icónica en el agente reactivo "perro buscador de trufas", para ello uso un vector bidimensional estático de 20*20 para asegurarme no salirme del mapa, también se podría haber empleado memoria dinámica en el caso que no conociera el tamaño máximo del mapa.

Esta solución propuesta, toma como pilar principal la matriz y será ésta la que guiará al agente a través del mapa, conforme éste lo va recorriendo y descubriendo:

- Detectar muros: en el caso que el agente choque contra un muro este quedara marcado con un valor predefinido en la constante MURO "1000". Esto evita que el agente choque más de una vez contra un muro, marcando el camino que recorrerá.

- Marcar la casilla por la que pasa: cada vez que el agente pasa por una casilla esta se incrementa en +1, esta información es utilizada para tomar la decisión de movimiento, el agente intentará ir hacia la casilla por la que menos veces a pasado por ella, comprobando para ello los valores de las casillas a su alrededor, arriba, abajo, izquierda, derecha.

1.2 Descripción de funciones:

Void RectiBump_( );

Función privada: se encarga de rectificar la posición tras el choque (es llamada cuando se activa la variable bump_) vuelve a la posición anterior, en función de la orientación que tenga en ese momento, distinta en cada orientación.

ActionType MejorElec( );

Función privada: se encarga de buscar la mejor posición posible entre las cuatro posibilidades, retorna ActionType.

Almaceno en un vector el valor que toma la matriz, arriba, abajo, izquierda y derecha con respecto a mi posición: primero compruebo si las cuatro posibilidades son iguales de buenas, en tal caso la acción elegida es avanzar.

Si no son iguales, localizo la posición mas ventajosa (la casilla por la que he pasado menos veces), en esta elección priorizo el avance por

encima de las demás posibilidades. Una vez que he localizado la mejor opción de desplazamiento, calculo la opción en función de mi orientación que almaceno en una variable aux;

ActionType Think( );

Función publica: encargada de tomar la decisión de movimiento apoyándose en las dos funciones privadas descritas anteriormente, retorna ActionType con la acción a realizar, reajustar la orientación y marcar los muros.

En la función tengo una estructura de control (switch), en la cual realizo acciones en función de la última decisión tomada (ult_accion_).

- Si la última vez fue avanzar (actFORWARD), reinicio a libre todas las posiciones del vector de obstáculos; si choco, marco en la matriz esa posición y en el vector de obstáculos, muro y true respectivamente, y llamo al método para que me ajuste la posición tras chocar.

- Si la última vez fue girar a la derecha (actTURN_R), reajusto la orientación.

- Si la última vez fue girar a la izquierda (actTURN_L), reajusto la orientación.

--Después aparece una estructura condicional if-else-if que primero extrae trufas (actEXTRACT) si el nivel oloroso es por encima de cinco.

--Si no hay, realiza la acción de oler, (actSNIFF).

--Si no se realizan las acciones anteriores, intenta buscar el mejor movimiento posible, evitando los muros, recalculando la acciones en función de la orientación e incrementando el valor de la casilla para usar esos valores en la decisión. En caso que haya delante un muro, el movimiento predefinido es girar a la derecha.

--En el caso que no entre en ninguna condición, el movimiento será hacia la izquierda.

Por último, igualo la última acción a la acción actual.

2. Resultados obtenidos por la solución aportada en los distintos mapas que se entregan con el agente.

MAPA	Average total collected "DoAllRum"
agent.map	1383
agent_rap.map	2383,2
mapa1.map	1437,4
mapa1_rap.map	2373,6
mapa2.map	1402,4
mapa2_rap.map	2284,6
mapa3.map	1365
Mapa2_rap.map	2247,4

3. Descripción de otras estrategias descartadas.

- Agente reactivo sin memoria icónica, descarté esta solución por el hecho de perder movimientos con los choques y no poder realizar una búsqueda heurística, hacia la mejor opción posible con el fin de recorrer la mayor cantidad de posiciones en el mapa.

- Uso de toma de decisiones aleatorias, he descartado todas las decisiones aleatorias que pueda tomar el agente, puesto que no se suele obtener una mejora constante.

Ejemplo:

En la función de buscar el mejor movimiento posible "MejorElec() ", en el caso que todas las casillas tengan igual valor, implemente un randon de acciones, pero después de diversas pruebas obtengo mejores resultados, si priorizamos el movimiento hacia delante.

- Nivel oloroso de la casilla “> 0” antes de extraer, después de varias pruebas, llego a la conclusión que si extraigo solo cuando el nivel oloroso es “> 5” en lugar de superior a cero, obtengo mejores resultados.

-Oler siempre antes de extraer, para intentar ahorrar iteraciones en el programa, probé a extraer siempre en lugar de oler y extraer, descartándolo por obtener peores resultados.