ROBOT CON PASILLOS ESTRECHOS

Asignatura: Sistemes Intel·ligents

Autor: Sebastià Vicens Oliver

1. INTRODUCCIÓN:

El principal objetivo de esta práctica es el desarrollo de un robot que recorre el perímetro de un objeto de forma que se permita la existencia de pasillos estrechos. Para ello, se hace uso de un estado interno del agente.

Para el mejor entendimiento del trabajo, es conveniente considerar un robot en un espacio bidimensional cuadriculado, limitado por una frontera y puede contener a su vez otros objetos de gran tamaño. El objetivo del robot es alcanzar una celda fronteriza y seguir su perímetro indefinidamente. En la *figura 1* podemos ver el esquema de forma gráfica.

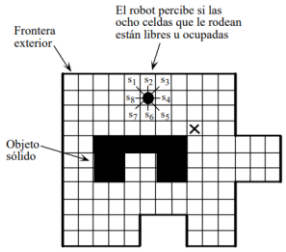

Figura 1. Esquema de la práctica, sacado del TALLER 2 (página, 17)

Los pasillos estrechos deben recorrerse íntegramente, sin que ello cause ninguna confusión al robot. Esto puede ocurrir si el robot confunde el objeto que está recorriendo y continua el recorrido a través de otro objeto.

En el caso de encontrarnos con un pasillo estrecho bloqueado, es decir, sin salida, como en el que indicamos en la siguiente figura, el robot debe recorrerlo íntegramente, dando a continuación la vuelta atrás con el objetivo de salir del mismo. En la *figura 2* encontramos un ejemplo de pasillo estrecho bloqueado y el recorrido que tiene que hacer el robot.

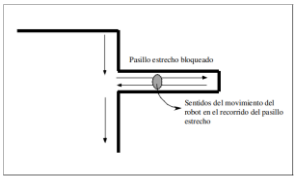

Figura 2. Pasillo estrecho bloqueado y recorrido del robot, enunciado práctica (pág. 3)

2. CARACTERÍSTICAS DEL AGENTE:

1.1. **Características de cada componente del REAS del agente:**

Para diseñar un programa de agente, es muy necesario contar con una idea bastante clara y precisa del **REAS** de un agente:

- Las posibles *percepciones*:

Para llevar a cabo el funcionamiento del robot, éste usará varias percepciones del entorno. Éstas son:

1. **W:** Vector que contiene la información sobre lo que tiene el robot en las casillas de su alrededor. El vector está forado por *[w0, w1, w2, w3, w4, w5, w6, w7]* y contiene la información de todos y cada uno de los cuadros que rodean el robot. En concreto, será 1 si hay una paret (el robot no puede passar por ahí), y 0 si no hay nada (el robot puede pasar).
2. **OldW:** Esta percepción también es otro vector (esta vez de 4 variables *[w0, w1, w2, w3]*) que indica lo que tuvo el robot en las posiciones *norte*, *sur*, *este* y *oeste* en la posición anterior.
3. **NewW:** Vector similar al *oldW* pero este contiene las posiciones *norte*, *sur*, *este* y *oeste* en la posición actual.
4. **ActS:** Adicionalmente poder realizar el recorrido de pasadizos estrechos, he utilizado este atributo, que indica la dirección del movimiento actual del robot. Si este se mueve hacia el *norte = 1, sur = 2, este = 3 y oeste = 4*.

- Las *acciones* que será capaz de adoptar el agente:

En este caso, el agente o robot, como se indica en el enunciado solo podrá llevar a cabo un tipo de acción. Éste es el desplazamiento en una unidad a través del entorno de juego (o cuadrícula). El desplazamiento podrá ser en uno de los cuatro sentidos disponibles: *norte, este, sur, oeste*.

- Las *metas* o medidas de desempeño que se supone debe llevar a cabo el agente:

Como se ha expuesto anteriormente, la principal meta u objetivo que debe llevar a cabo el agente es el recorrido del perímetro de un objeto de forma que se permita la existencia de pasillos estrechos. Dicho recorrido será realizado en sentido horario.

- El tipo de *ambiente* en qué tal agente operará:

El ambiente se trata de un espacio bidimensional cuadriculado, limitado por una frontera y puede contener a su vez objetos de gran tamaño que simulan muros por los cuales el agente no puede pasar.

1.1. Estado interno del agente. Qué variables forman parte del mismo y cómo se asocian dichas variables.

Como se ha expuesto en la descripción de los componentes del *REAS*, en concreto, en la descripción de las percepciones, el agente cuenta con 5 variables distintas. Para una mejor explicación de éstas, recomiendo pegar un vistazo en el punto anterior. En este punto vamos a ver como podemos asociar dichas variables.

Tres de las cuatro variables que forman parte del estado interno del agente tienen una gran relación. Bueno, en realidad, todas, pero vamos a ver la relación de los tres vectores **W**, **oldW** y **newW**. En este caso, tenemos tres vectores para describir el conjunto de paredes que rodea el robot debido a que hemos supuesto que el robot solo podrá conocer las características de los posiciones (o cuadros) que tiene en el norte, este, sur, y oeste: ***[w1, w3, w5, w7]***. Por tanto, para obtener el resto de características *[w0, w2, w4, w6]* se utilizará el vector **oldW** que dependiendo del movimiento anterior que haya ejecutado el agente, podremos conocer dos de las características del vector **W**. Es decir, si el robot se mueve hacia el norte, se conocerá la información del actual *[w1, w3, w5, w7]* y, además utilizando el vector **oldW**, podremos obtener las características *sureste* (*w4*) y *suroeste* (*w6*). Mientras que las posiciones (*w0, w2*) se mantendrán a 0 aunque haya alguna pared.

Adicionalmente, encontramos la variable **actS** que nos indica el sentido actual del agente. Éste también está muy relacionado con los otros vectores, ya que si el agente se mueve hacia el norte, el vector **W** se tendrá que actualizar dependiendo de los vectores **oldW** y **newW**.

Para poder realizar el recorrido a través de los pasadizos estrechos se usa la variable llamada **actS** que nos permite recorrer los pasadizos exitosamente.

1.2. El vector de características. Es decir, el conjunto de variables que se usarán en el antecedente de las reglas. La descripción debe ser en pseudocódigo y muy clara.

En este punto se mostrará como el agente obtiene las percepciones del ambiente. Para esto hay que conocer cual es el esquema interno que debe seguir el agento para su funcionamiento. Éste es:

función *AGENTE (percepción):*

mientras (Verdadero):

antecedente ← interpretar_percepciones(percepción)

movimiento ← razonar_posible_movimiento(antecedente)

realizar_movimiento(movimiento)

f_mientras

f_función

Como vemos es un bucle infinito donde se realizan tres acciones muy diferenciadas. La interpretación de las percepciones, el razonamiento del posible movimiento a realizar, y finalmente, la realización del movimiento. En este punto nos centraremos en el primer punto, la interpretación del conjunto de características o variables que se usarán en el antecedente de las reglas. Por tanto, la función *interpretar_percepciones* es la siguiente:

función *interpretar_percepciones(percepción):*

var *newW ← [0, 0, 0, 0], oldW, W ← [0, 0, 0, 0, 0, 0, 0, 0], actS, oldS*

inicio

actualiza_W(percepción, oldW, newW, W)

f_función

La función actualiza_W actualiza el valor de los vectores ***oldW***, ***newW***, y ***W***:

función *actualiza_newW(percepción, oldW, newW, W):*

inicio

newW[0] ← ParedN(percepción)

newW[1] ← ParedE(percepción)

newW[2] ← ParedO(percepción)

newW[3] ← ParedS(percepción)

por *i* ***en*** *rango(4):*

*W[i*2+1] = newW*

si *actS == 1:*

uniónN(oldW, W)

sino si *actS == 2:*

uniónS(oldW, W)

sino si *actS == 2:*

uniónE(oldW, W)

sino si *actS == 4:*

uniónO(oldW, W)

oldW ← newW

f_función

Las funciones *uniónX* lo que hacen es añadir las dos percepciones que el agente ha inferido de la posición anterior. Por ejemplo, la función de *uniónN* sería:

función *unión(oldW, W):*

inicio

W[4] ← oldW[1]

W[6] ← oldW[3]

f_función

1.3. **La base de reglas. Las reglas deben estar dispuestas en orden descendiente según su prioridad de ejecución. Estas reglas deben describirse en pseudocódigo para que sean fácilmente interpretables. No se admitirá un listado del programa fuente.**

En este apartado se mostrarán las reglas usadas para poder realizar el movimiento del agente en base a sus percepciones. Estas reglas están implementadas en las funciones *razonar_posible_movimiento* y *realizar_movimiento* expuestas en el punto anterior.

La función *razonar_posible_movimiento* tiene como parámetro el vector ***W*** y la variable ***actS***, y sería de la siguiente forma:

función *razonar_posible_movimiento(antecedente)* ***retorna*** *acción:*

inicio

si *W[1]* ***and no*** *W[3]* ***and*** *(actS != O* ***or*** *(W[7]* ***and*** *W[5]))* ***and*** *(****no*** *(actS == N* ***and*** *W[6]* ***and no*** *W[7])):*

return *ESTE*

sino si *W[3]* ***and not*** *W[5]* ***and (****actS != N* ***or*** *(W[1]* ***and*** *W[7]))* ***and not*** *(actS == E* ***and*** *W[0]* ***and not*** *W[1]):*

return *SUD*

sino si *W[5]* ***and not*** *W[7]* ***and*** *(actS != E* ***or*** *(W[3]* ***and*** *W[1]))* ***and not*** *(actS == S* ***and*** *W[2]* ***and not*** *W[3]):*

return *OESTE*

sino si *W[7]* ***and not*** *W[1]* ***and*** *(actS != S* ***or*** *(W[5]* ***and*** *W[3]))* ***and not*** *(actS == O* ***and*** *W[4]* ***and not*** *W[5]):*

return *NORTE*

sino si *W[0]* ***and not*** *W[1]* ***and*** *actS != S:*

return *NORTE*

sino si *W[2]* ***and not*** *W[3]* ***and*** *actS != O:*

return *ESTE*

sino si *W[4]* ***and not*** *W[5]* ***and*** *actS != N:*

return *SUD*

sino si *W[6]* ***and not*** *W[7]* ***and*** *actS != E:*

return *OESTE*

sino:

si not *W[1]:*

return *NORTE* *# Vamos al norte por defecto*

si not *W[3]:*

return *ESTE* *# Vamos al este por defecto*

si not *W[5]:*

return *SUR* *# Vamos al ser por defecto*

si not *W[7]:*

return *OESTE* *# Vamos al oeste por defecto*

f_función

Finalmente, en la función de realizar movimiento se realiza el movimiento dependiendo de la percepción que se reciba de la función descrita anteriormente. Además actualiza el valor de la variable ***actS*** según el movimiento que realiza el robot:

función *realizar_movimiento(movimiento):*

inicio

mover(movimiento) *# Esta función cambia la posición del robot*

actS ← movimiento *# Se actualiza el valor de actS*

f_función