INTELIGENCIA ARTIFICIAL

Practica 1: Agentes reactivos.

Documentación del comportamiento del agente.

Photo By: kazuend on unsplash.com

Francisco Navarro Morales - GRG121

Segundo curso del Grado de Ingeniería Informática
Universidad de Granada
curso 2016-2017

1. Movimiento.

El agente ha utilizado tres métodos para elegir el próximo movimiento. El primero consiste en mantener una matriz 'memoria' de enteros, inicializada a cero e ir sumando uno a cada casilla que el agente pise [fig. 2]. Así, las casillas con más valor son aquellas por las que se ha pasado más veces. El otro método es similar pero no suma uno a la casilla que pisa, sino que se le da el valor de un 'contador' o reloj, que empieza a 0 y va aumentando en cada turno [fig. 1]. Así, no se controla cuantas veces se ha pasado por una casilla, sino que se controla cuáles son las últimas casillas por las que se ha pasado. La combinación consiste en utilizar el segundo método pero en lugar de asignar el valor del contador a la casilla, se le suma. De forma que a la hora de comparar casillas se tiene en cuenta cuál es la última que se ha pisado y, además, cuantas veces se ha pisado [fig. 3].

Fig 1

0	0	9	0	0
6	7	14	13	12
0	0	3	0	0
0	0	2	0	0
0	0	1	0	0

Fig 2

0	0	1	**0**	0
1	2	4	2	1
0	0	1	**0**	0
0	0	1	**0**	0
0	0	1	**0**	0

Fig 3

0	0	9	0	0
6	12	36	24	12
0	0	3	0	0
0	0	2	0	0
0	0	1	0	0

Así, si nos fijamos en las figuras y, teniendo en cuenta que a la hora de decidir qué dirección tomar solo se comparan los valores de enfrente, la derecha y la izquierda de dónde se encuentra el agente, y se elige el menor, y suponiendo que el agente mira hacia arriba en la casilla remarcada: en el caso de la izquierda el agente giraría a la izquierda; en el del centro, avanzaría; y en el de la derecha (el que he escogido) avanzaría.

Hay que tener en cuenta que el agente antes de pensar si quiera en girar a la derecha, la izquierda o avanzar, comprueba si dicha acción es posible (si no hay algún obstáculo que se lo impida). Así, en el ejemplo que he puesto, si en la casilla de arriba de dónde se encuentra el agente hubiera un obstáculo, en la segunda opción cogería cualquiera de las dos, izquierda o derecha, aleatoriamente; o bien, una establecida por defecto; mientras que en el caso de la derecha se giraría a la izquierda porque es por la que se pasó hace más tiempo de las dos.

Aunque en un principio me pareció que el método que combinaba los otros dos era mejor, pero tras hacer pruebas vi que el método que solo comprueba cual se pisó menos recientemente da mejores resultados y decidí utilizar ese.

También es interesante señalar el uso de una función que establece casillas que no me interesa pisar, de forma que les asigna el valor actual de contador de tiempo más diez, de forma que durante los próximos turnos sea la última opción a elegir. Un ejemplo del uso de esta función es cuando el agente encuentra un lobo y no tiene ningún hueso que darle, entonces le asigna dicho valor a la casilla del lobo y así se obliga a sí mismo a girar y "huir"del lobo.

Además he introducido un factor 'aleatorio' que obliga al agente a girar a la casilla mejor (izquierda o derecha) según el criterio establecido aunque la casilla del frente tuviera más prioridad y los resultados han sido buenos ya que, por ejemplo, cuando el agente (por la disposición de los obstáculos) realiza

una trayectoria rectangular cerrada, tiende a repetir dicha trayectoria interiorizando en el rectángulo (haciéndolo cada vez más pequeño hasta haberlo pisado entero) y esto es bastante absurdo porque en cada vuelta al rectángulo solo va rellenando una casilla del borde cuando su visión le permitiría rellenar más. Al introducir el giro aleatorio se impide que de demasiadas vueltas a estos rectángulos (entre otras cosas). He escogido que se gire obligatoriamente cada veintitrés pasos tras probar a darle varios valores y ver que los mejores resultados se generaban con 42 y 23 pasos (con 23 eran un poco mejores así que elegí ese número).

2. Objetos

Durante las sesiones de prácticas he 'descubierto' que los huesos se pueden dar a los lobos para hacerlos desaparecer durante un par de turnos con la acción actGIVE, el bikini sirve para poder pisar las casillas de agua, las zapatillas para las casillas de bosque y la llave para abrir puertas (también con actGIVE). He tenido en cuenta que no se puede cambiar las zapatillas por el bikini para pasar de bosque a agua, o al revés, porque en el momento en que el agente se quita el objeto que lleva y le permite pisar la casilla en la que está, muere. En un principio programé que cuando viera un objeto que ya tenía se lo equipara y lo tirara para coger el nuevo pero esto provocaba que hubiera pocos objetos en el mapa, por eso lo cambié a utilizar la función NoPisar sobre la casilla con el objeto para evitarlo. También hubiera sido interesante poder dejar el objeto en el suelo a un lado y luego coger el nuevo, pero por alguna razón me ha sido imposible hacer que el agente deje los objetos en el suelo (he probado todas las circunstancias que se me han ocurrido con todos los objetos y no los suelta), por lo que he concluido que era imposible (quizá por algún fallo en el programa Belkan o simplemente porque yo sería incapaz de encontrar las circunstancias requeridas a tiempo) y me he limitado a esquivar el objeto que ya tengo.

Para 'equipar' un objeto necesario para pisar un determinado terreno, abrir una puerta o eliminar a un lobo, lo que hace el agente es ver si tiene el objeto (para lo que mantiene un vector de booleanos con los objetos que posee y los que no) y, si lo tiene, comprueba si lo lleva encima, si no, comprueba si tiene algún otro objeto para guardarlo (actPUSH) y si no lleva ninguno coge algún objeto de la mochila (actPOP).

3. Mapa

He programado que todo lo que el agente ve se introduzca en el mapa resultado y además, he establecido una matriz auxiliar para ir guardando lo que ve cuando aún no está situado e introducirlo cuando se sitúa. También he introducido una matriz de memoria auxiliar para utilizar el método de movimiento cuando aún no se está situado. Para situarse antes, he programado un módulo que hace que, si se ve una casilla de GPS frente al agente, se le obligue a avanzar aunque la mejor opción fuera girar y, si en la memoria se ha guardado una de estas casillas a izquierda o derecha del agente y las casillas que hay entre el agente y la casilla GPS se puden pisar, entonces gira en dicha dirección.

4. Puertas

Por último para asegurarme de que se abren las puertas, si el agente lleva una llave y tiene una puerta a la derecha o izquierda, gira obligatoriamente para mirar hacia ella, entonces la abre y pasa. El problema es que esto hace que una vez dentro vuelva a girar para abrirla y salir y no entra en la sala (e incluso se

queda atascado). Por ello he creado una variable entera ‘abierta’ que se pone a 5 cuando se abre la puerta y se va restando en una unidad cada turno desde entonces de forma que no se intente abrir la puerta otra vez si la variable es mayor que cero.