

INTELIGENCIA ARTIFICIAL

Practica 2:

Combinando comportamientos reactivos y deliberativos.

Documentación del comportamiento del agente.



Photo By: joanna-kosinska-44214 on unsplash.com

Francisco Navarro Morales - GRG121

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

1. Como agente reactivo.

Aunque he reutilizado gran parte del código de la práctica anterior, he modificado muchos aspectos del lado reactivo del agente para obtener mejores resultados. Por ejemplo, he eliminado el comportamiento reactivo que tenía mi agente para abrir puertas siempre que tenía una llave y una puerta a uno de los lados, lo he sustituido por la capacidad de abrir puertas como parte de planes que requieran entrar a ellas, y luego ejecuta la salida como un plan que lleva a un objetivo imaginario que se crea cuando se pasa la puerta en la posición inmediatamente anterior a la puerta(desde donde se entra). He añadido que el agente suelte todos los objetos que tiene poco antes de morir y he modificado la forma en que se actualiza el vector que contiene información sobre qué objetos tiene el agente en cada momento: la idea es poner el elemento correspondiente al objeto que porta a false siempre que realice la acción actTHROW o actPUTDOWN y, en el turno siguiente, comprobar si el objeto activo es nulo y, de no ser así, poner a true el booleano puesto que significaría que no ha sido posible lanzar el objeto. Un aspecto bastante interesante que he modificado del lado reactivo de mi agente es que he añadido una nueva capa a la parte que decide si girar o seguir adelante para sustituir al método de ir acumulando en una matriz "memoria".^{el} último turno en que se pisó cada casilla y moverse con preferencia por las casillas que se pisaron menos recientemente (o el método de ir sumando uno a la casilla que se va pisando e ir a las que han sido pisadas menos veces). Mi idea ha sido que, dado que el objetivo del agente es **rellenar todo el mapa, y no pisar todo el mapa**, el criterio que debía seguir, preferentemente, era qué movimiento me desvelaba más casillas por descubrir. Por tanto he programado otra matriz similar a la matriz "memoria" que ya utilizaba en la que sumo uno a cada casilla que ha sido vista en algún momento por el agente y además, tengo una función que determina la suma de los valores de las casillas que serán vistas por el agente si decide avanzar, girar a la izquierda o girar a la derecha. Así, en cada momento comprueba qué movimiento será el que produce un valor menor (y por tanto, el que tiene casillas que han sido vistas menos veces). Esto es especialmente útil para evitar que el agente haga ciclos innecesarios rodeando casillas que **no ha pisado** pero **sí ha visto**. No he hecho un estudio exhaustivo de si este método mejora o empeora los resultados porque en la mayoría de los casos el porcentaje de mapa visto fluctuaba en función de si daba la casualidad de que encontraba un objeto determinado antes o después (ya que objetos como las botas o el bikini permiten recorrer el mapa mucho más deprisa) pero aún así los resultados son muy buenos y la impresión que da el agente al ejecutarse es mucho mejor (en mi opinión) que como estaba antes. Otro detalle es que debido al tamaño de la mochila, que permanece como en la práctica anterior, y que con el añadido de los regalos es insuficiente para llevar todos los objetos a la vez, para no complicarme he decidido que el agente no coja nunca el hueso.

2. ¿Cuando planeo?

En primer lugar, y principalmente para las etapas del comienzo o después de una muerte, mi agente recoge un conjunto de posiciones con objetos que quiere y no tiene en cuanto los ve y, en cuanto puede, hace un plan para ir a por ellos (Lo he hecho con una búsqueda en anchura porque la implementé antes de implementar el algoritmo de A estrella y me pareció interesante usarla en cortas distancias por ser algo más rápida y sencilla y porque quería darle algún uso). Además de esto, la planificación de ir a por regalos se hace si no se tienen regalos y, para ello, ordeno todos los regalos de los sensores en función de la distancia Manhattan al estado actual e intento ir a por los 3 primeros. Hace lo mismo con los reyes, si tiene regalo, ordena un conjunto de posiciones dónde sabe que hay reyes (si los ha visto) e intenta ir a por los más cercanos. En ambos casos, si no puede encontrar un camino, para evitar que siga buscando en el próximo turno, he programado que pase 30 turnos usando la parte reactiva sin planear.

3. Algoritmos de búsqueda

En primer lugar hice una implementación de la búsqueda en anchura, que mejoré para convertirla en el algoritmo A estrella pero que mantuve para distancias cercanas porque consideré que sería más eficiente. En ambos casos he utilizado listas con las acciones que llevan a un estado concreto y, para mejorar la eficiencia en tiempo y facilitar la programación, he utilizado como estructura de datos una cola con prioridad como frontera y un unordered map para almacenar los costes menores para llegar a un nodo y el subplan menor necesario para llegar a dicho nodo. Como estado utilice tanto la posición como la orientación, de forma que los hijos de un nodo sean el resultado de avanzar o girar a algún lado. He puesto **un límite de 600 iteraciones de búsqueda** para evitar que haga búsquedas demasiado largas, aunque dado que también hay un límite de distancia antes de iniciar la búsqueda, dudo que se complete demasiado. La heurística de la búsqueda es la distancia manhattan entre los nodos intermedios y objetivo. Además, algo interesante que he añadido es, cuando se va a introducir en la frontera un nodo que procede de la acción de avanzar, compruebo que no ocurra que el nodo actual y el nodo al que avanza sean uno agua y el otro bosque porque entonces no sería posible avanzar aunque tuviera tanto bikini como zapatillas. En tal caso, el nodo no se añade a la frontera. Además, para evitar que escoja caminos que necesiten equipar y desechar las botas y el bikini varias veces, he añadido un poco más de coste a los nodos que requieren equipar un objeto (es decir, los que pasan de un terreno normal a agua o bosque).