+

Universidad De Granada

Grado ingeniería informática (2016 – 2017)

Inteligencia Artificial

Práctica 2 | Agente reactivo + Pathfinding

Trabajo realizado por: Antonio Miguel Morillo Chica

+

1. Introducción

En la primera práctica diseñamos un agente puramente reactivo con memoria, es decir, reaccionaba al mundo del agente a través de sus sensores, además posee una representación icónica del mundo por lo que guardaba todo lo que ve más información para saber cuando estuvo en un lugar determinado.

En esta segunda práctica buscamos crear encontrar y crear caminos para satisfacer misiones/objetivos, así, nuestro agente se convertirá en una especie de agente proactivo con memoria.

2. Comportamiento reactivo.

El comportamiento reactivo de mi agente existen tres funciones básicas para este tipo de comportamiento, una actualiza el estado del mundo según la información previa y sensores y por otro lado las otras dos comprueban lo que hay justo delante y justo debajo.

La función comprobarJustoDelante toma decisiones para reaccionar tanto al terreno como a la superficie, el flujo de control continua sobre la variable "decisión", estos módulos (funciones) cambian el valor de la variable para indicar que hay que hacer manejos con la mochila y/o con la superficie.

Por otro lado está comprobarJustoDebajo que controla, por un lado, si el robot está ubicado o no para así poder pintar en el mapa resultado y por otro lado el volcado sobre aux_muerto y rescate a aux_matrix.

2.1. Toma de decisiones.

Existe una fuerte sintonia en mi código entre dos de las funciones señaladas arriba. Todas ellas continúan con el flujo de acción sobre la variable "decisión". Finalmente el modulo comprobarCasilla decidirá gracias a la representación icónica del mundo y de los sensores, decide si debe o no evitar cierto obstáculo.

2.1.1 Matriz auxiliar, reconociendo el mapa.

Uno de los puntos más fuerte del agente reactivo es el hecho de construir en memoria una representación del mapa por el que me estoy moviendo, es decir, recordar el mapa que voy viendo. Una vez ubicado, es decir, cuando se ejecuta el

+

método comprobarJustoDebajo() con éxito el mapa en memoria pasará directamente al mapa resultado.

Además de la matriz auxiliar uso una matriz muerto para guardar los cambios vistos antes de morir en cada vida para así poder rescatar toda esta información cuando el agente se vuelve a ubicar en la siguiente vida y en todas las demás.

2.1.2. Actualizar información.

El módulo actualizarInformación mantiene actualizada la representación del mundo para el agente según las acciones que realiza. Una vez ubicado mantiene actualizadas la ubicación real y relativa del agente para así poder construir en todo momento caminos.

Por otro lado este método usa la guardarQueso que inicialmente se pensó para guardar en todo momento la visión del terreno y superficie en todo momento. Esto ha cambiado con el desarrollo. El módulo mencionado guarda el queso tanto en el mapaResultado como en nuestra aux_matrix pero en la visión de la superficie cambia. Solo guardamos los reyes que vemos y la entida justo delante para que el modulo comprobarCasilla lo detecte.

Nota: Debería haber usado el sensor en vez de guardar la entidad, no me ha dado tiempo a modificarlo.

3. Comportamiento deliberativo.

Debido a la introducción de elementos nuevos en la práctica se nos incita a llevar a cabo comportamientos deliberativos, es decir, usar la información del mundo necesaria para poder construir una secuencia de acciones que nos llevarán a un destino determinado.

3.1. Cuando ejecutar deliberativo.

Hay distintas necesidades que debemos cumplir para poder ejecutar un comportamiento reactivo, tener objetos necesarios para realizar la travesía, es decir, poder cambiar de objetos si el camino es necesario para cambiar entre los tipos de superficie, haber visto a algún rey y estar ubicado.

Hay dos formas de comenzar a ejecutar un plan:

- Ir a por objeto: El plan se ejecutará solo si el agente sabe o ha visitado la ubicación del objeto. Esto lo hago porque el comportamiento reactivo funciona muy bien y como acaba explorando la mayor parte del mapa por inercia acabará ejecutando un plan de este tipo.
- Dar un objeto: El plan se ejeccutará solo si se ha visto al menos un rey. Aún así es necesario poseer los objetos para la travesía.

El módulo que decide cuando realizar o no un camino es buscarPlanificarEjecutar que se encarga de decidir cuando ejecutar un plan y si debe o no construir uno nuevo.

3.2. Implementación A*.

La implementación de mi método de búsqueda ha sido un algoritmo de tipo primero el mejor conocido por A*. Las listas de abiertos y cerrados son un priority_queue y un set de nodos y estados, respectivamente. Un nodo contiene un estado completo de un nodo, los costes y además una lista de las acciones necesarias para llegar al nodo.

La implementación del A* se ve en la función pathFinder además esta se ayuda de insetarHijos para descartar obstáculos o hijos que pueden perjudicar la eficiencia.

Hay tres condiciones de parada para el a*, que el nodo mejor de abiertos no sea un nodo peor que el dole del coste de ir desde el inicio al goal, que abiertos se quede vacío y que se encuentre una buena solución.