Entregable 1 SID JADE
Q2-2023

Priyanka Amarnani Karamchandani, Andrés Emch Boada, Marc Sayós Casasas

1. Descripción del problema

Se busca resolver un problema en el entorno Dedale que involucra a dos agentes, A y B, que comienzan en posiciones aleatorias del mapa y deben intercambiar sus posiciones en algún momento de la ejecución. Al inicio de la ejecución, A no conoce la posición de B, y viceversa, pero ambos pueden comunicarse a través del método sendMessage con un rango de comunicación configurado en 1. El agente B no puede moverse hasta que tenga conocimiento de la posición del agente A, y se pueden presentar intersecciones relevantes que puedan causar bloqueos en el movimiento de los agentes.

2. Decisiones de diseño

Para empezar nuestro diseño de la solución, hemos decidido adaptar una implementación de la clase ExploSoloBehaviour que realiza un DFS por todo el mapa recorriendo todos los nodos sin dejar ninguno sin visitar, al no tener la capacidad de comunicarse con el otro agente a menos que esté a un rango de distancia de 1 va enviando mensajes de forma cíclica. El agente B está esperando dicho mensaje para devolver una respuesta de confirmación, al recibir dicha confirmación A finaliza su búsqueda y finaliza así nuestro SearchBehaviour. El mensaje que recibe el Agente B se compone de diferentes partes: por un lado dos localizaciones, la origen del agente A y la final del agente A y, por otro lado, un grafo serializable que representa el mapa que ha ido descubriendo en su búsqueda.

Al recibir dicha información, agente B se dispone a desplazarse hacia la ubicación inicial del agente A, para ello, aprovechará la función de GetShortestPath que nos proporciona el map del agente, y que junto con la localización destino emprenderá la ruta óptima para llegar a la posición inicial del agente A. Cómo al principio ese path es muy probable que le lleve por la posición en la que se encuentra el agente A, busca entre los nodos vecinos y empieza moviéndose por uno de ellos. De esa forma deja espacio para que el agente A llegue a su destino. Seguidamente sigue buscando nodos entre sus vecinos hasta que encuentra uno ya explorado por el agente A, y a partir de ahí usa la función GetShortestPath para llegar a su destino.

3. Distribución del trabajo

- Agente explorador: Priyanka
- Agente recolector: Andrés
- Search behavior: Marc