Campus Santa Fe

Fecha de Entrega: 15/11/2023

Materia: Modelación de Sistemas Multiagentes con Gráficas Computacionales (Gpo 302)

Actividad. Roomba

Domingo Mora | A01783317

El problema en cuestión es el siguiente; un robot conocido como “Roomba” el cual tiene la función de limpiar espacios. Al comenzar su nueva aventura el roomba no conoce su ambiente, su único conocimiento son los movimientos que puede hacer, donde se ubica su estación de carga y ya. Dentro de la configuración del robot, tiene varios factores que permiten que este robot pueda servir sin la necesidad de un controlador humano. Para desarrollar este agente aleatorio primero se debe definir el ambiente donde va a ser colocado este “Roomba”. Este ambiente se conforma de objetos predeterminados como: “sillas, mesas, macetas” o algún tipo de obstáculo que puede encontrar el robot dentro del cuarto. También existe la posibilidad de un objeto indeterminado como un humano, mascota entre otras cosas que se ingresan al ambiente sin estar ahí previamente, pero por el momento trabajaremos con lo determinado. Finalmente el “Roomba” como definición limpia espacios, por lo cual también se necesitan basuras colocadas aleatoriamente en cada simulación.

Comencemos con los factores predeterminados que son los siguientes:

- Habitación MxN espacios
- Numero de Agentes
- Cada acción cuesta 1% de la batería
- Basura colocada aleatoriamente
- Regreso a estación de carga (También como carga)

Se necesitan medir ciertos valores a lo largo de la simulación para poder ver la efectividad del trabajo del roomba en cuestión del tiempo que tardó en limpiar, los pasos que se tardó y el porcentaje de celdas que están limpias.

Comencemos con la primera simulación:

Esta primera simulación tiene solo 1 roomba que se inicializa en la coordenada {1,1}, al igual que su estación de carga. Primero tenemos que definir cómo se va a ver el autómata dentro del script del servidor, aquí defino los colores que toma cada elemento (Basura, roomba, y obstáculos), junto con la cantidad de elementos que se van a colocar en el grid. Finalmente las gráficas de los datos almacenados se colocan aquí para poder verse; este script también corre el autómata. Luego seguimos al modelo: En este script se inicializa el autómata, dentro de este script se colocan los obstáculos, basura, agentes (Dentro de los agentes se tiene que revisar las posiciones que se pueden utilizar). Dado a que los obstáculos y basura se colocan de forma aleatoria se necesitan dos funciones que revisen las posiciones y otra que las regrese para que se puedan colocar. Finalmente el agente, en este script se define la funcionalidad del roomba y como va actuar dentro del autómata. Se necesitan componentes para quitar basura del grid cuando el autómata pasa sobre esa posición, también para definir los extremos del grid para que no actúe fuera del área predeterminada. El autómata también tiene batería, para la estación de carga se define en el modelo pero la acción para regresar a esta posición en caso de que tenga que recargar cabe dentro del agente, donde se usa el algoritmo de a-star que viene dentro de la librería "networkx". Este algoritmo lo que hace es que sepa el camino a la posición donde se encuentra la estación sin tener todas las posiciones almacenadas. La librería de "networkx" también nos da una función llamada "shortest_path" que ayuda para buscar la basura dentro del grid.

Las características del ambiente son las siguientes:

- Agentes (ObstacleAgent, RandomAgent, TrashAgent)
- Visualizacion de los Agentes
    - Radio 0.5
    - Agentes con diferentes colores
        - Obstacle: Gris (Layer 1)
        - Random: Azul (Layer 2)
        - Trash: Amarillo (Layer 3)
    - Parámetros
        - N : Agentes
        - M : Obstáculos
        - T: Basura
        - Altura
        - Ancho
    - Grid
        - 15 x 15 celdas
        - 500 x 500 pixeles en el display
    - Servidor en el puerto 8521

Estadísticas:

- Tiempo necesario para que todas las celdas estén limpias (Cuánto tardo)
- Porcentaje de celdas limpias

- Número de movimientos realizado por el agente

En conclusión, el programa modela eficazmente un entorno simple basado en agentes con obstáculos y agentes dinámicos que interactúan en una cuadrícula. El uso de la biblioteca Mesa agiliza el proceso de implementación, haciéndolo accesible para crear y visualizar simulaciones basadas en agentes. Si bien se podrían realizar ciertos ajustes y mejoras para escenarios más complejos, el programa proporciona una base sólida para experimentar con comportamientos de agentes en un entorno simulado. Fuera de lo mencionado toda la logística del autómata funciona de igual manera.

Simulación 2:

En la segunda simulación los parámetros cambian ligeramente; para empezar, en esta simulación existen varios agentes que empiezan en posiciones aleatorias; donde también se ubican su estación de carga. Los agentes conocen solamente la posición de su estación de carga inicial, pero pueden cargarse en cualquier estación de carga. Estas son las únicas modificaciones que se le hacen al autómata.

Las estadísticas ahora cambian para cada agente dependiendo de cómo actúa individualmente en la simulación. Sin embargo los datos que se están recuperando son los mismos

Estadísticas:

- Tiempo necesario para que todas las celdas estén limpias (Cuánto tardo)
- Porcentaje de celdas limpias
- Número de movimientos realizado por el agente

Las características del ambiente son las siguientes:

- Agentes (ObstacleAgent, RandomAgent, TrashAgent)
- Visualizacion de los Agentes
    - Radio 0.5
    - Agentes con diferentes colores
        - Obstacle: Gris (Layer 1)
        - Random: Azul (Layer 2)
        - Trash: Amarillo (Layer 3)
    - Parámetros
        - N : Agentes
        - M : Obstáculos
        - T: Basura
        - Altura
        - Ancho
    - Grid
        - 15 x 15 celdas
        - 500 x 500 pixeles en el display

- Servidor en el puerto 8521

El modelo de autómata modificado refleja un escenario más realista y desafiante, donde los agentes deben navegar dinámicamente, explorando el entorno mientras administran estratégicamente sus recursos energéticos mediante el uso de múltiples estaciones de carga. La adaptabilidad y autonomía de los agentes contribuyen a un proceso de limpieza más sólido y eficiente, ofreciendo información valiosa sobre la dinámica de los sistemas autónomos en escenarios del mundo real.

Arquitectura de subsunción: