Tecnológico de Monterrey

Actividad MA: Roomba

Modelación de sistemas multiagentes con gráficas computacionales (Gpo 302)

Alumno

S Fernanda Colomo F - A01781983

16 Nov 2023

Sylvia Fernanda Colomo Fuente -A0178198

Problemática

Para esta actividad se nos solicitó la simulación del funcionamiento una *roomba* la cual es definida como “un robot aspiradora autónomo” (Team, 2023). Para poder implementarlo se usó la librería de Mesa en Python para el manejo de agentes, modelo y servidor donde se visualizará la simulación.

Agentes en la simulación

RandomAgent (Roomba)

1) Objetivo: Limpiar las celdas “sucias”(DirtyAgent) moviéndose por el grid y cargándose cuando sea necesario (es decir volver a su estación de carga).Cabe aclarar que no es omnisciente
2) Capacidad Efectora: La roomba puede moverse en cualquier dirección disponible (es decir no obstaculizada), puede cargarse (saber cuando es necesario cargar su batería), limpiar las celdas y elegir la celda a moverse revisando posibles obstáculos.
3) Percepción: El agente puede ver las celdas que lo rodean (es decir sus vecinos) así puede detectar celdas con obstáculos, cargadores, otras roombas y suciedad.
4) Proactividad: La roomba siempre le dará prioridad a limpiar el espacio por lo que siempre que tenga batería disponible priorizará ir hacia cualquier celda sucia detectada en vez de moverse a un espacio vacío.
5) Métricas de desempeño: Se calculará entre el tiempo que le tome a la roomba(s) terminar de limpiar el tablero dado. También se puede calcular la batería usada en comparación a las celdas sucias existentes por cada vez que el roomba vuelve a su estación de carga.
6) Reactividad: Es reactivo ya que si se presenta algún cambio como la presencia de una celda sucia o falta de batería se hará la acción correspondiente.
7) Habilidad Social: Como tal no hay intercambio de información entre los agentes por lo que la única interacción que hay es cuando se limpia la celda o se comparte celda con un cargador.

DirtyAgent

1) Objetivo: Representar una celda sucia que debe ser limpiada por una roomba.
2) Capacidad Efectora: Se elimina cuando se encuentra con una roomba.
3) Percepción: Sólo percibe si comparte celda con una roomba. Pero no se comunica con su entrono.
4) Proactividad: –
5) Métricas de desempeño: Se calculará la cantidad de instancias de este agente en un tiempo determinado.

6) Reactividad: Es un leve reactivo ya que se elimina cuando detecta que en su celda hay una instancia de un (RandomAgent)
7) Habilidad Social: Como tal no hay intercambio de información entre los agentes solo si llega a interactuar con un RandomAgent, se elimina esa instancia.

ObstacleAgent

1) Objetivo: Tiene como objetivo representar un obstáculo a evitar cuando el roomba se mueve.
2) Capacidad Efectora: –
3) Percepción: –
4) Proactividad: –
5) Métricas de desempeño: Se podría hacer una estimación de la eficiencia al limpiar con y sin obstáculos.
6) Reactividad: –
7) Habilidad Social: –

Como tal no cuenta con las demás características del diseño de un agente dado que solo se coloca al inicio de la simulación pero no hace nada. El que debe estar al pendiente de si hay algún agente de este tipo cerca es el RandomAgent. Por eso su función de step se le coloca un pass.

ChargerAgent

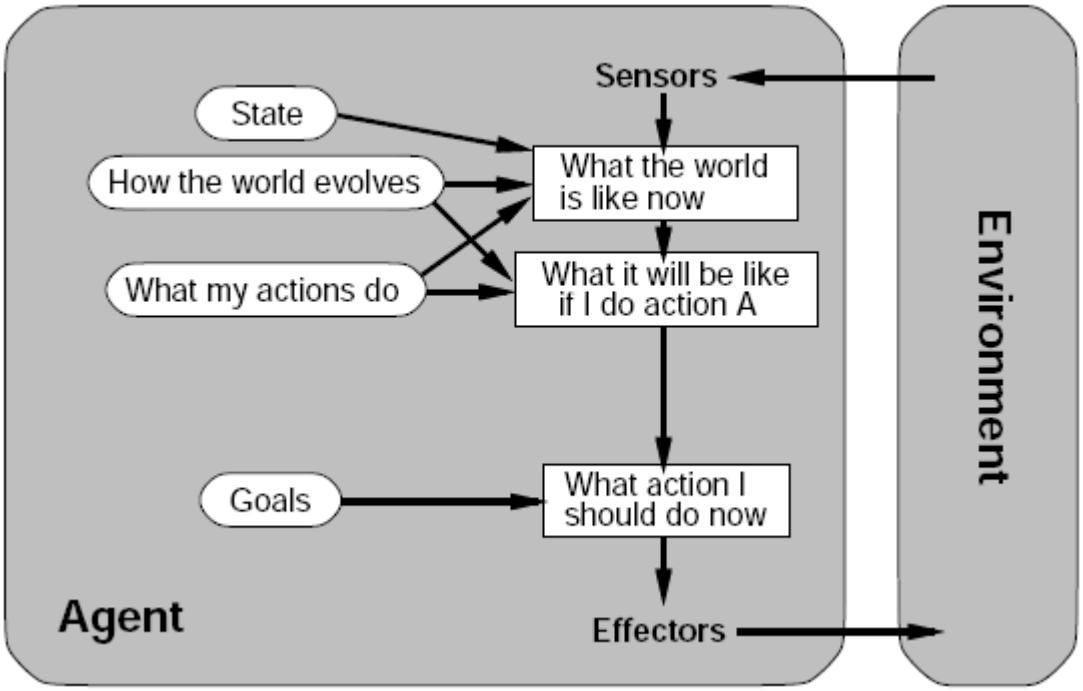
1) Objetivo: Tiene como objetivo representar un obstáculo a evitar cuando el roomba se mueve.
2) Capacidad Efectora: –
3) Percepción: –
4) Proactividad: –
5) Métricas de desempeño: –
6) Reactividad: –
7) Habilidad Social: –

Como tal no cuenta con las demás características del diseño de un agente dado que solo se coloca al inicio de la simulación pero no hace nada. El que debe estar al pendiente de si hay algún agente de este tipo cerca es el RandomAgent para poder cargarse. Por eso su función de step se le coloca un pass.

La arquitectura de subsunción de los agentes

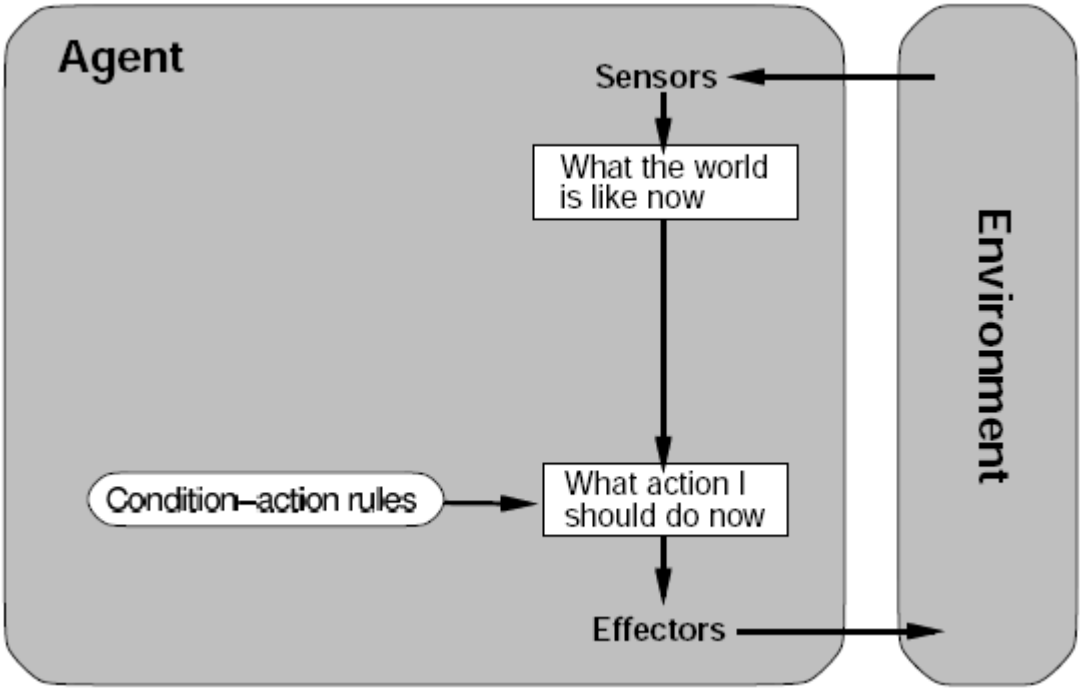
Diagramas de arquitectura de subsunción de cada agente

RandomAgent (Roomba)



Se eligió esta arquitectura dado que el RandomAgent ya que este tiene una meta que es limpiar el espacio dado. Para esto tiene sensores que le permiten reconocer su entorno que lo rodea, pero únicamente las celdas adyacentes a él mismo. Siempre revisa los agentes que hay en esas celdas y toma la acción con mayor prioridad. Por ejemplo si hay alguna celda sucia la va a limpiar o si hay un obstáculo y no hay celdas sucias elige otra celda para moverse. Igual también revisa su nivel de batería con el fin de volver a cargarse.

DirtyAgent



Se eligió esta arquitectura dado que el agente tiene que ver si comparte celda con un RandomAgent y si lo hace debe eliminarse, de lo contrario no hace nada.

ObstacleAgent

Dado que el agente no hace ninguna acción, no tiene arquitectura, simplemente es creado al inicio de la simulación.

ChargerAgent:

Dado que este agente tampoco tiene alguna acción no cuenta con arquitectura. Recordemos que el RandomAgent es quien se encarga de percibir si está sobre un ChargerAgent y si lo está sigue la acción de cargarse. Como tal en la simulación y código entregado ese es el comportamiento, si de lo contrario se modificará el comportamiento en el código para que el cargador aumente la batería de el RandomAgent se hablaría de una arquitectura de reflejo simple como la del (DirtyAgent)

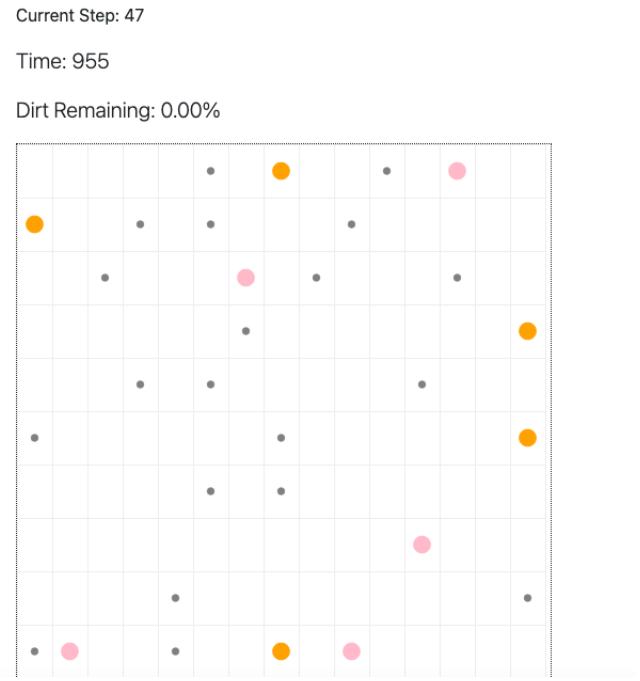
Características del ambiente

Se cuenta con un ambiente inaccesible dado que el agente no puede acceder a toda la información del mismo, ya que se encuentra limitado al conocimiento de sus celdas adyacentes únicamente. Es un ambiente determinista dado que todas las acciones cuentan

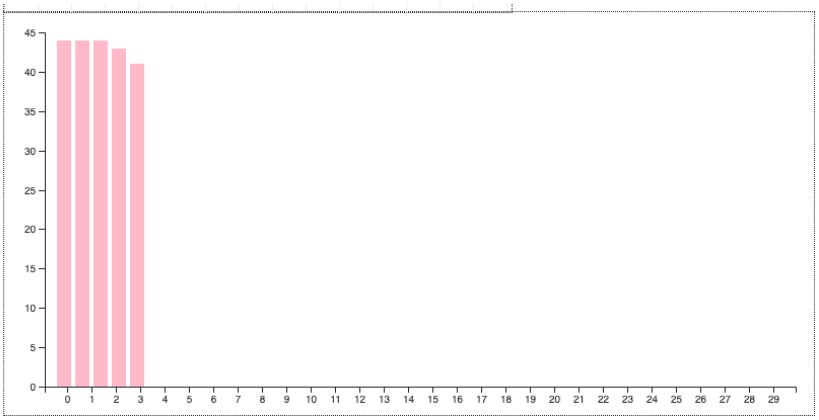
con su resultado esperado tales como la eliminación de un DirtyAgent, el movimiento y carga de un RandomAgent.A su vez es un agente episódico discreto dado que cuenta con un número finito de acciones, por ejemplo tiene un tiempo máximo de ejecución y se tienen movimientos finitos y fijos de acciones como limpiar y moverse. Finalmente se cuenta con un ambiente estático dado que este permanecerá sin cambios significativos que vayan fuera del control de las acciones dadas en cada agente.

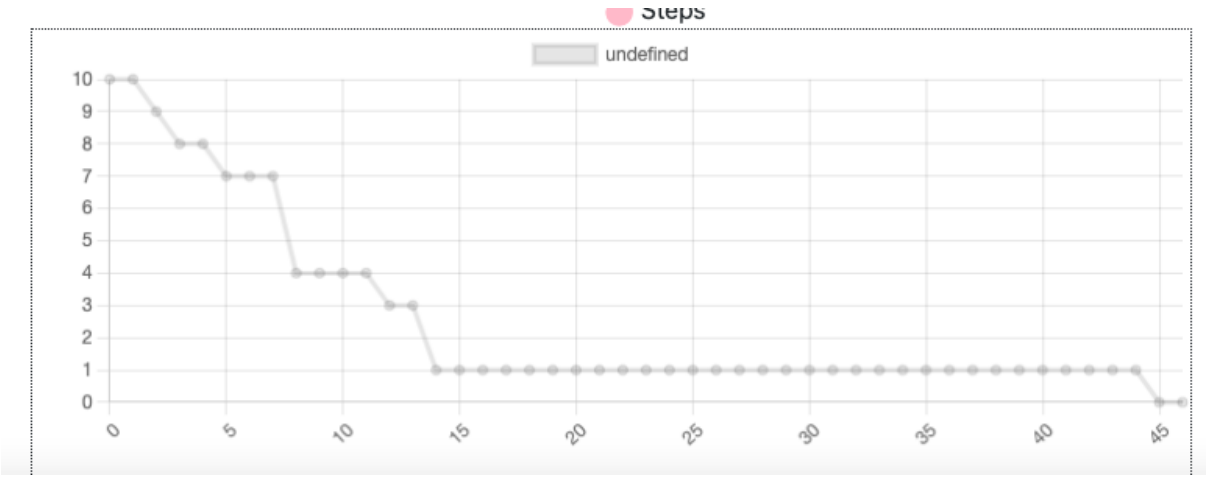
Estadísticas recolectadas:

5 RandomAgent, 10 ObstacleAgent y 10 DirtyAgents



Sylvia Fernanda Colomo Fuente -A0178198



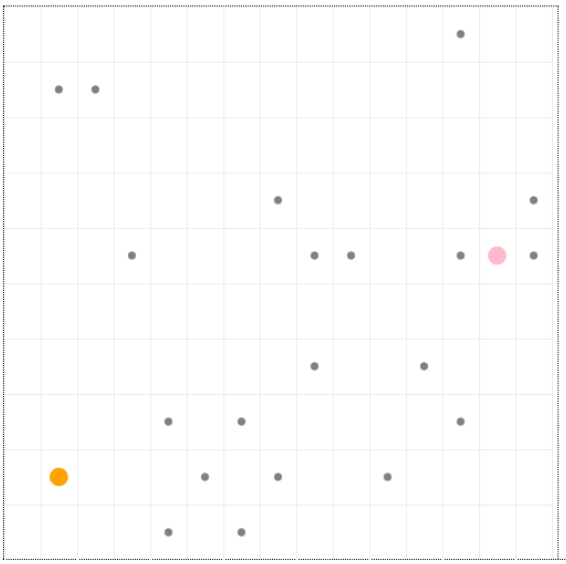


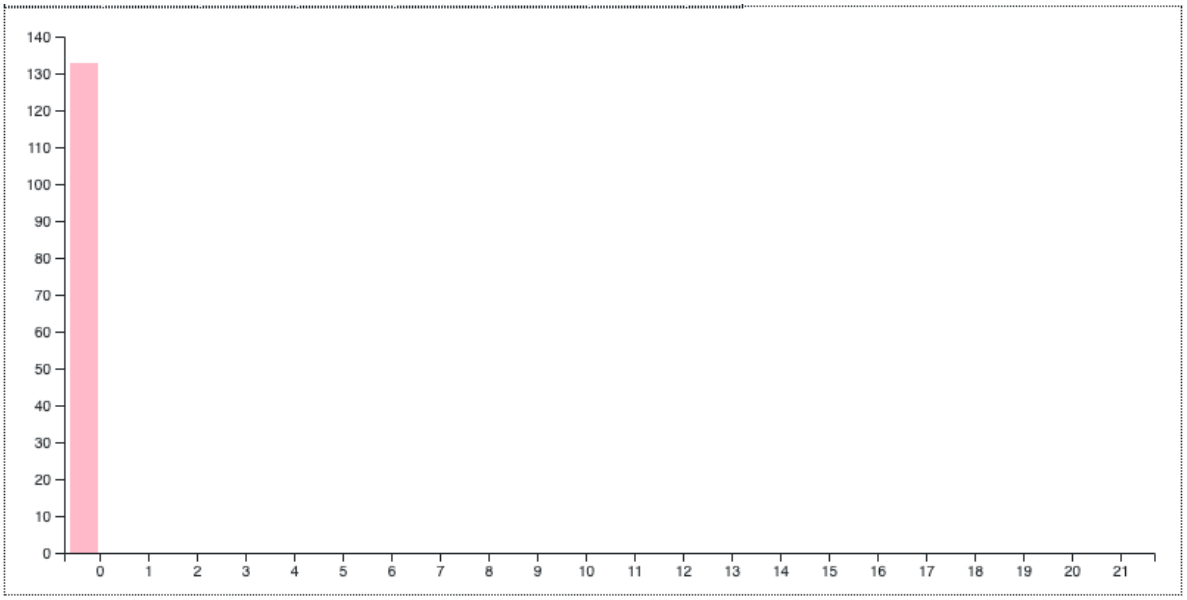
1 RandomAgent, 10 ObstacleAgent y 10 DirtyAgents

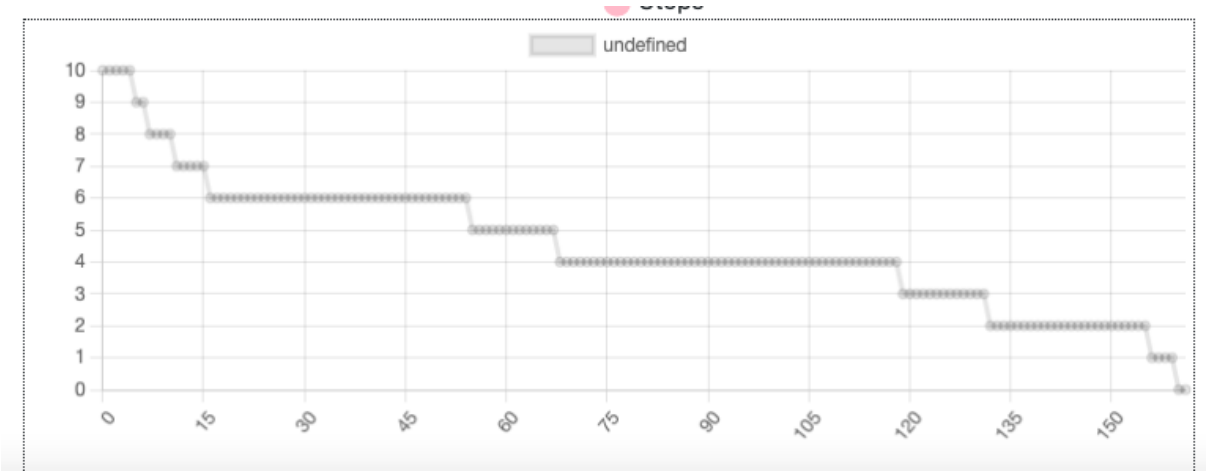
Current Step: 162

Time: 840

Dirt Remaining: 0.00%







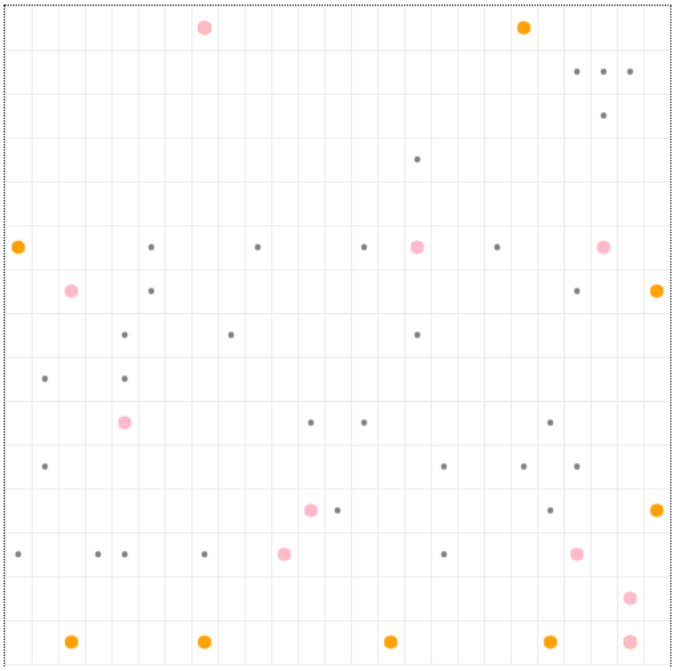
10 RandomAgent, 30 ObstacleAgent y 20 DirtyAgents

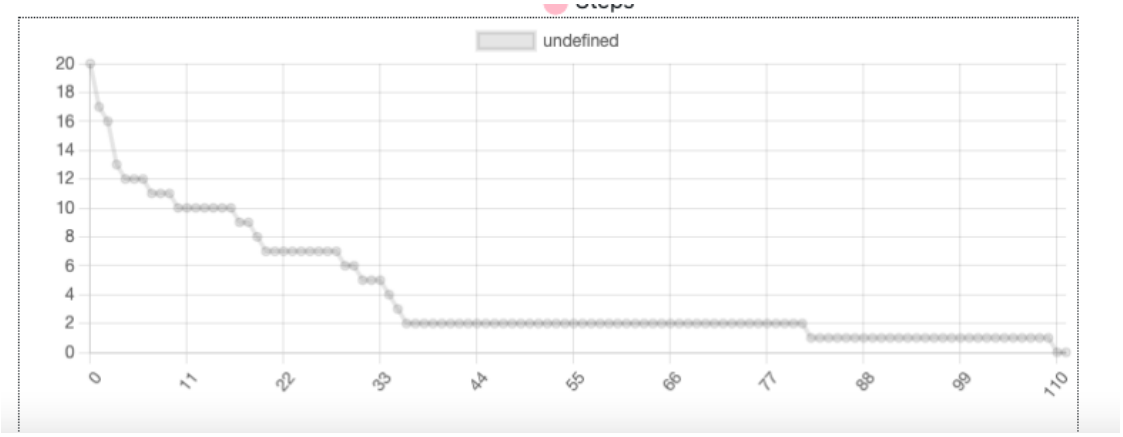
25 x 15 grid

Current Step: 112

Time: 890

Dirt Remaining: 0.00%





Conclusiones

Después de analizar estas tres simulaciones podemos ver que un agente es mucho menos eficiente que un número de agentes mayor para limpiar el mismo espacio, incluso es posible que los agentes no se carguen cuando en la simulación de un mismo agente si se acaba cargando al menos una vez. En la simulación con cinco agentes podemos esperar que se vea más seguido en la gráfica la disminución de basura cuando en la otra gráfica tenemos una irregularidad. Finalmente en una simulación con un grid y mayor cantidad de agentes podemos ver el tablero se acabó de limpiar el el step 110 mientras que en la simulación con un agente y un grid más pequeño se acabó en el 140, lo cual nos lleva a pensar que debido a estos resultados el número de roombas respecto al grid era pequeño, por lo que se debe considerar mejor la cantidad de roombas. Por ejemplo pudimos ver que para el grid pequeño una cantidad de cinco roombas es bastante eficiente pero por ejemplo si se agregaran 10 roombas si sería eficiente pero gastaría más recursos por lo que es importante analizar lo obtenido y buscar la cantidad óptima de roombas por el espacio dado.

Cabe aclarar que la limitación del movimiento de los agentes al igual que se sabe que son omniscientes pues retrasa la simulación pues pueden visitar espacios no necesariamente útiles o tomar caminos largos para encontrar celdas sucias por lo que incluso se puede llegar a ver una simulación con un agente el cual no acabe de limpiar en el tiempo dado. Por lo anterior se recomendaría en futuras implementaciones que se cambiara el movimiento a algo no aleatorio con un pathfinding.

Finalmente este ejercicio me fue útil para comprender el funcionamiento de los agentes y cómo en base a diferentes estímulos pueden llevar a cabo diferentes acciones. También aprendí sobre el nivel de complejidad que tiene la tecnología integrada hoy en día dado que esta roomba no tiene el mismo comportamiento que una verdadera. Fue una gran experiencia el poder manipular un servidor con sus debidos agentes no sólo para entender su comportamiento sino obtener datos estadísticos de los mismos y llegar a una conclusión personal y numérica.

Referencias

Team, R. (2023, September 8). *Roomba*. ROBOTS: Your Guide to the World of Robotics. https://robotsguide.com/robots/roomba