Tecnológico de Monterrey

ITESM Campus Santa Fe

Bloque:

Modelación de sistemas multiagentes con gráficas computacionales (Gpo 301)

Entregable:

MA. Actividad: Roomba

Diego de la Vega Saishio - A01420632

Profesor:

Octavio Navarro Hinojosa

Fecha de entrega:

19 de noviembre de 2025

Problema

La situación abordada consiste en la simulación de la limpieza de una habitación por robots roombas. Cada uno de ellos cuenta con una estación de carga y tiene la tarea de limpiar la habitación lo más rápido posible, evitando obstáculos.

Solución propuesta

A través del uso de sistemas multiagentes, se generarán aleatoriamente las rumbas solicitadas en el mapa, un porcentaje de obstáculos del 10% y de basura del 20%. Si se selecciona una roomba nada más, ésta se genera en la celda (1,1) con su estación de carga respectiva, mientras que si son más, se generan aleatoriamente en el mapa. Con esto empezamos la segunda simulación, en donde, inicialmente, las roombas solo conocen su propia estación de carga, pero al ir perdiendo energía, se aseguran de tener la suficiente para poder volver (con 10 de energía como margen de error). El método para encontrar este camino es BFS (en las funciones "calculate_path_station" y "calculate_path_to_work"), que a través de un deque permite encontrar el camino más corto en grafos sin pesos. Utilicé ambas funciones porque la primera es para dirigirse a la estación de carga que tenía en un inicio (aunque si se encuentra otra en el camino, la utiliza) y la segunda es para que regrese a la posición en la que ya estaba trabajando. Esto evita buscar áreas ya analizadas con anterioridad. Además, es importante mencionar que, si una roomba utiliza una estación de carga que no era la suya y llega la que sí le pertenece, hice que esperara a que se desocupe para cargarse, ya que no esperar no consume energía.

A esto, le añadí un tiempo máximo de 1000 steps, en donde se detiene la simulación, sin importar que se haya limpiado todo el cuarto o no. Por otro lado, si logran limpiar el cuarto antes de este tiempo, se detiene la simulación.

Nota importante:

Con el fin de evitar crear dos carpetas, 1 para simulación 1 y la otra para la 2, cree un pequeño menú desplegable para elegir la simulación. La primera limita las roombas a 1, en la celda (1,1), mientras que la segunda sí puede tener 1 o más, pero se generan aleatoriamente en el mapa. Cabe aclarar que, aunque la simulación 1 cuenta con las funcionalidades de la 2, estas no sirven por ser solo 1 roomba (ej. puede buscar otras estaciones de carga, pero no hay más que la suya).

Diseño y características de los agentes

Si bien los agentes, en principio, serían los roombas, los obstáculos, la basura y las estaciones de carga, estos 3 últimos no se comportan como agentes realmente, pues solo se generan aleatoriamente, y el/los roomba(s) toman decisiones a partir de su presencia. Por esta razón, las características a evaluar serán únicamente las de las roombas:

1) **Objetivo:**
    a) Limpiar la habitación y recargar batería
    b) Si bien se busca que esto sea eficiente y en poco tiempo, su objetivo principal es limpiar la basura que se encuentra y no quedarse sin batería en el proceso.
2) **Capacidad efectora:**
    a) **Moverse** en 8 direcciones (vecindario)
    b) **Limpiar** basura en su celda actual
    c) **Cargar** batería al estar en una estación de carga
    d) **Esperar** sin consumir energía cuando una estación está ocupada
3) **Capacidad de percepción:**
    a) **Percepción alrededor:** Vecinos
    b) **Percepción global limitada:** Solo conocen el camino de vuelta a su estación de carga a través de BFS.
    c) **Estado interno:** Conocen su nivel de energía actual y la distancia que tienen hacia su estación.
    d) **Memoria:** Conocen la posición en la que estaban trabajando antes de ir a cargarse.
4) **Proactividad:** Implementé un comportamiento proactivo en las roombas con las funcionalidades solicitadas y unas extra:
    a) Planificación: Calculan el momento en el que deben regresar a su estación de carga, con un margen de error de 10 de energía.
    b) Ruta más corta: Uso del algoritmo BFS para encontrar el camino más corto a la estación de carga.
    c) Jerarquía de tareas: Busca activamente celdas con basura a su alrededor para recolectarlas, prioriza la limpieza, pero al tener poca batería, busca una estación de carga.

d) Ubicación de trabajo: Guardan su punto de trabajo antes de cargar, para eficientar la búsqueda.
e) Comportamiento oportunista: Aprovechan estaciones de carga en su camino de vuelta para optimizar tiempo.

5) **Reactividad:** Tienen una radiactividad alta, pues perciben la basura a su alrededor y actúan en consecuencia, al igual que con las estaciones de carga y obstáculos. Por otro lado, si la estación de carga que le pertenece está ocupada, espera.
6) **Habilidad social:** No implementé ninguna funcionalidad social, pero para el futuro, podrían darse a conocer entre ellas ubicaciones de estaciones de carga, obstáculos y celdas visitadas para eficientar la búsqueda.
7) **Racionalidad:** Los clasificarías con una racionalidad alta, pues son capaces de prever acciones para el futuro. Por ejemplo, para la carga de energía, la re-ubicación en su zona de trabajo anterior, y la espera a que se desocupe una estación de carga.
8) **Métricas de desempeño:**
    a) Trash: Se mide la cantidad de basura restante para evaluar el porcentaje de limpieza.
    b) Movement: Movimientos realizados para alcanzar su objetivo.
    c) Energía: Sirve para evaluar qué tan bien administró su energía para maximizar la eficiencia y evitar que se descarguen los roombas.
    d) Tiempo: Los pasos que le toma limpiar un área, además de un límite de tiempo.
9) **Omnisciencia:** No son omnisicientes por las siguientes razones:
    a) Conoce celdas vecinas y su origen (estación de carga)
    b) No conocen toda la distribución de la basura, obstáculos y estaciones de carga.
10) **Aprendizaje:**
    a) Nulo: No aprende de sus decisiones y/o consecuencias de sus acciones.
11) **Autonomía:** Son bastante autonómos los roombas, pues una vez que se crean, cumplen su funcionalidad sin que se les tenga que hacer algo para que recarguen o sigan buscando basura.

Arquitectura de subsunción de los agentes

Implementación a través de un modelo jerárquico de comportamientos y decisiones, donde cada nivel tiene una superioridad o inferioridad frente a los demás:

1) Nivel 1: Supervivencia
    a) Este comportamiento tiene la mayor prioridad, pues si su energía llega a cero, el agente muere.
2) Nivel 2: Recarga
    a) Si su energía es menor a 100 y está en una estación de carga, se tiene que cargar el roomba.
    b) Si está ocupada la estación, espera a que se desocupe.
3) Nivel 3: Movimiento
    a) Si está con 100 de energía en su estación de carga:
        i) Empezar a buscar basura
        ii) Regresar a su estación de trabajo anterior
    b) Si necesita cargarse, regresar a su estación de carga
4) Nivel 4: Limpieza
    a) Busca celdas contiguas con basura
    b) Si no hay celda contigua con basura, se mueve aleatoriamente
    c) Recolecta la basura de la celda a la que se movió

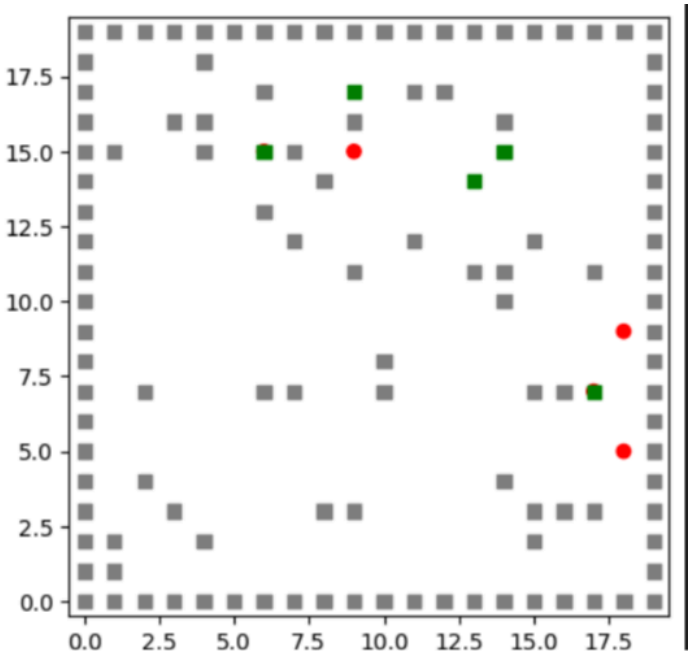
Características del ambiente

1. Parcialmente accesible: El ambiente solo es parcialmente observable por los agentes, pues las roombas perciben su vecindario inmediato y conocen su propia estación de carga, pero no todos los obstáculos, estaciones de carga ni basura.
2. Determinista: Las acciones en el ambiente tienen consecuencias predecibles y consistentes. Si se recoge basura, ésta no vuelve a aparecer, los obstáculos no se mueven y los agentes siempre pierden una energía al moverse o limpiar.
3. No episódico: Las decisiones actuales tomadas en el ambiente afectan su estado futuro de manera acumulativa. Por ejemplo, cuando una roomba recoge basura, ésta desaparece permanentemente.

4. Semi-dinámico: Clasifico esto así porque la basura solo aparece una vez y no cambia el estado del ambiente, a menos que un roomba recolecte una basura.
5. Discreto: Número finito de acciones, de celdas, de obstáculos, basura y, por supuesto, de agentes (roombas).

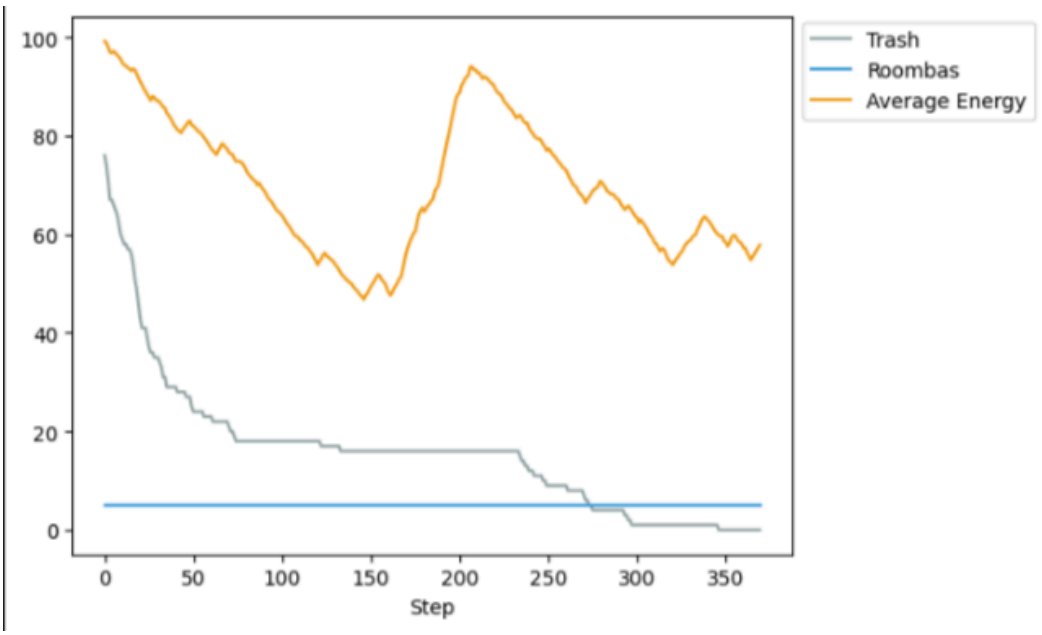
Estadísticas recolectadas en las simulaciones

Mediante el DataCollector de mesa, se recolectaron las siguientes estadísticas (se muestran con un ejemplo):



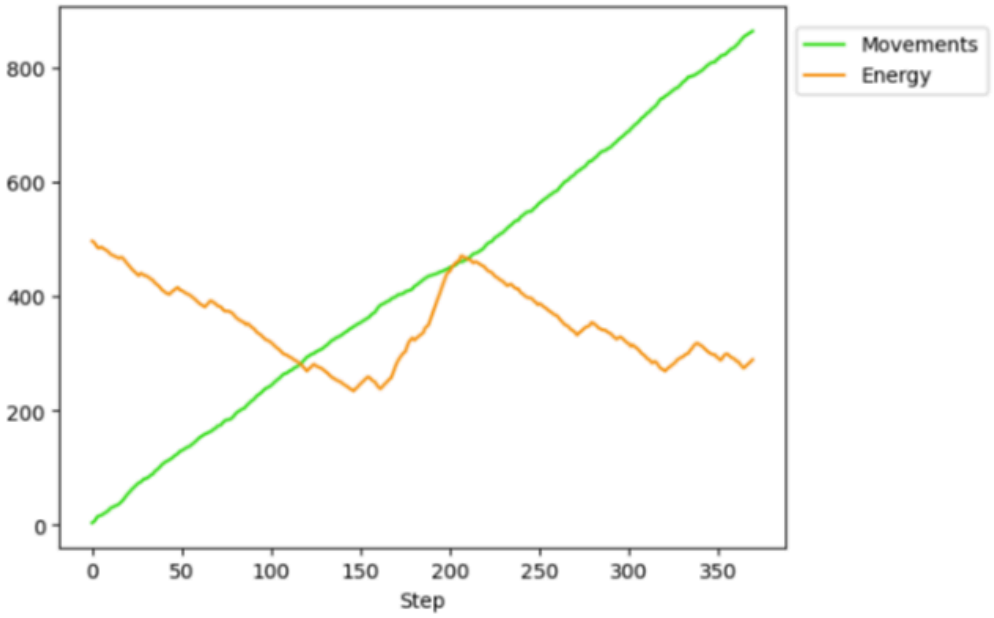
Primer gráfica:

1) Trash (basura restante): Utilizado para ver el comportamiento y la eficiencia de limpieza de los roombas conforme al tiempo.
2) Roombas: Utilizado simplemente para demostrar que ninguna roomba muere (se queda sin batería) en el transcurso de la simulación.
3) Energía Promedio: Utilizado para ver cómo se comportan las roombas y se cargan conforme se quedan sin energía. Al poner un margen de error de 10 de energía, se puede ver que el promedio no baja de alrededor de 50.



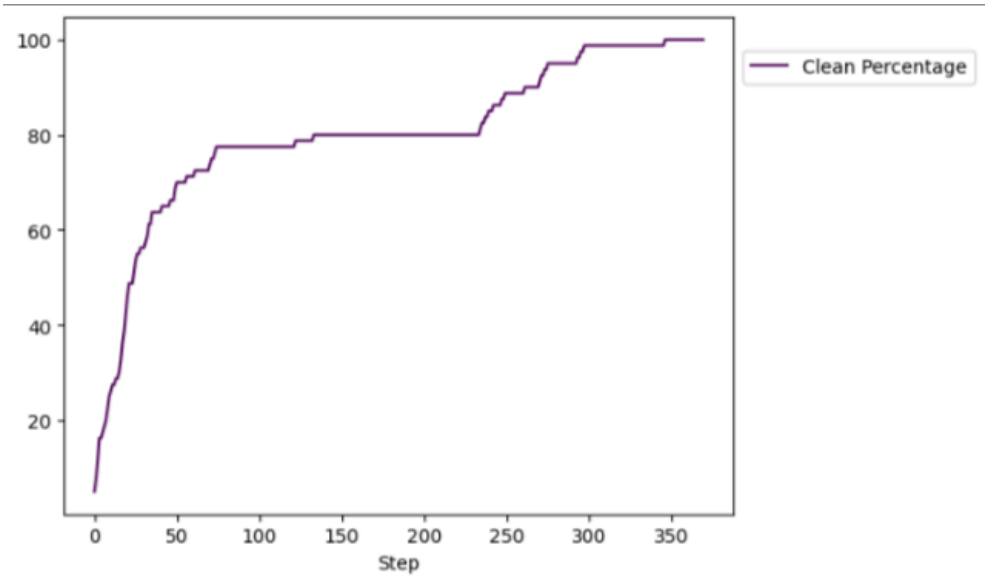
Segunda gráfica:

1) Movements (movimientos totales): Utilizados para visualizar si la energía no reduce cuando el agente está detenido. Se muestra un comportamiento inversamente proporcional con la energía (hasta la primera carga).
2) Energía: Se suma toda la energía de las roombas para ver el consumo energético de todas. Es más útil visualizarlo con un solo roomba para evaluar la proporción inversa anteriormente mencionada.



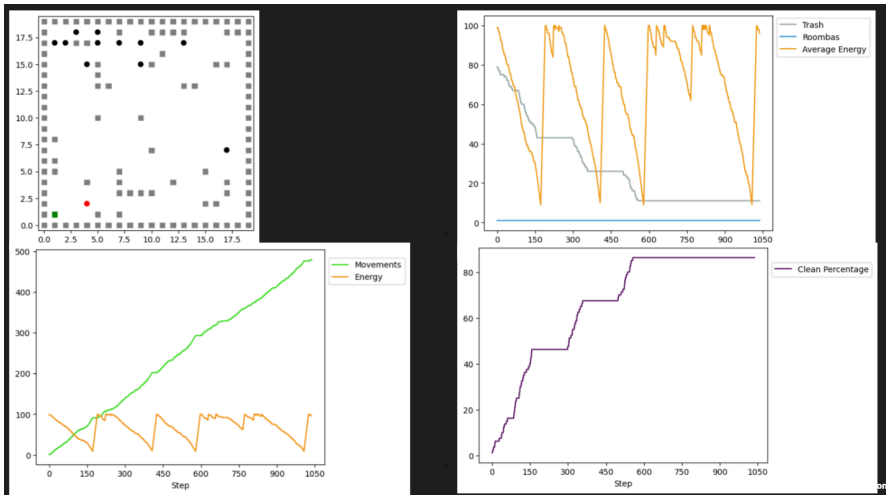
Tercera gráfica:

4) Porcentaje limpio: Medido con fines más ilustrativos para ver la recolección de basura con respecto a la habitación.



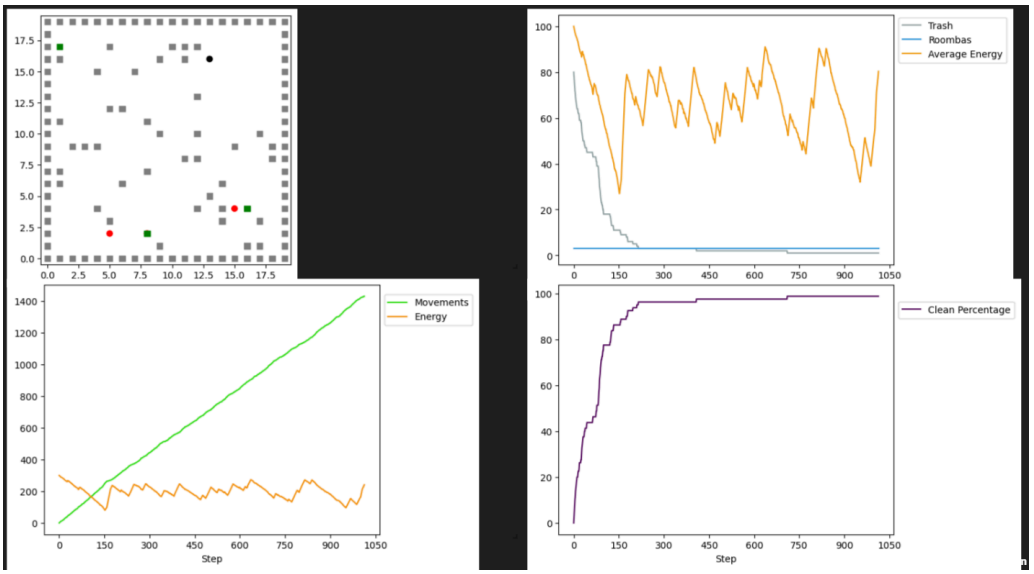
Conclusiones

La simulación presentada en la sección anterior fue con 5 agentes, realizando la misma con solo 1 agente, se puede observar que el tiempo no es suficiente para limpiar toda la habitación:



Si bien casi logra su objetivo, el hecho de que este en un borde del mapa, complica la exploración de todo (de manera aleatoria) y, por tanto, su limpieza. Además, el mínimo de energía con la que se queda es 10, el margen de error mencionado anteriormente. Los movimientos en todos los casos son incrementales, pero si le quitáramos el límite de 1000 pasos, esta segunda prueba daría resultados aún más elevados que la primera prueba.

Para una última prueba, se utilizarán 3 agentes:



Se puede observar que aún así solo quedó una basura restante y se utilizaron muchos más movimientos en conjunto. Sin embargo, es necesario tomar en cuenta la aleatoriedad para llegar a una celda que tenga basura vecina.

Los métodos que se podrían utilizar para mejorar este desempeño serían guardar la memoria para que eviten celdas visitadas y se dediquen los agentes a explorar nuevas. Por otro lado, se podrían implementar habilidades sociales para que entre ellos se comuniquen las estaciones de carga conocidas y las zonas exploradas (basura y obstáculos). Esto permitiría un trabajo en equipo entre los agentes y una mayor eficiencia en el desempeño de la simulación.