MA. Actividad: Roomba

Profesor: Octavio Navarro
Materia: MA
Nombres:

- Amilka Daniela Lopez - A01029277

Fecha: 18 de noviembre de 2025

Introducción

El presente reporte tiene como objetivo analizar y simular el comportamiento de un sistema de agentes tipo Roomba dentro de un ambiente discreto representado por una cuadrícula. Para ello se desarrollaron dos simulaciones: una con un solo agente y otra con múltiples agentes interactuando entre sí. Cada Roomba opera bajo una arquitectura de subsunción, tomando decisiones basadas tanto en su percepción local como en su estado interno, principalmente su nivel de batería. Los agentes deben navegar, evitar obstáculos, limpiar celdas sucias, regresar a estaciones de carga y, en la versión multiagente, cooperar compartiendo información del entorno. A lo largo de la simulación se registran métricas clave como cantidad de suciedad removida, movimientos y batería, lo que permite evaluar el desempeño de los agentes y el impacto del trabajo distribuido en comparación con un único robot autónomo.

Desarrollo

a) Problema a resolver

El problema a resolver consiste en simular un sistema de agentes autónomos encargados de limpiar una habitación representada como una cuadrícula de tamaño M×N. El entorno contiene celdas sucias, obstáculos y estaciones de carga, y los agentes deben recorrer el espacio para limpiar la mayor cantidad posible de celdas antes de que se agote el tiempo máximo de ejecución o su batería. El reto principal radica en que cada acción realizada consume energía, por lo que los agentes deben tomar decisiones que equilibren su capacidad de exploración con la necesidad de regresar oportunamente a una estación de carga. Adicionalmente, en la simulación con múltiples agentes, deben coexistir y desplazarse de forma eficiente dentro del mismo entorno, evitando choques o inconsistencia en la lógica de su comportamiento y distribuyendo la tarea de limpieza.

b) Propuesta de solución

La propuesta de solución se basa en el diseño de agentes autónomos con arquitectura de subsunción, capaces de percibir su entorno inmediato, reaccionar a la presencia de suciedad u obstáculos y planificar la recarga de energía cuando sea necesario. Para optimizar la navegación y la toma de decisiones, los agentes utilizarán algoritmos de búsqueda BFS (Breadth-First Search) para localizar de manera eficiente las celdas sucias más cercanas y las estaciones de carga disponibles de tal forma que puedan alcanzar su objetivo sin quedarse sin energía. Se implementa una lógica de navegación que permite al agente desplazarse, limpiar y gestionar su batería de forma autónoma, priorizando tareas según su estado interno y las condiciones del entorno. En la versión multiagente, cada agente opera con las mismas reglas pero con percepción local, compartiendo estaciones de carga cuando es necesario. La simulación recopila métricas como porcentaje de limpieza, movimientos realizados y tiempo requerido, lo que permite evaluar el desempeño individual y colectivo, así como analizar el impacto de aumentar el número de agentes en la eficiencia total del sistema.

c) Análisis de los agentes

1. **RandomAgent**

El RandomAgent es un agente móvil y proactivo, cuyo objetivo es limpiar la mayor cantidad posible de celdas sucias dentro del grid, evitando obstáculos, gestionando su batería y

regresando a la estación de carga cuando lo necesite. En términos de PEAS, su Performance measure se basa en celdas limpias, evitar quedarse sin batería y mantenerse operativo; su Environment es el grid con obstáculos, suciedad, estaciones y otros agentes; sus Actuators son el movimiento por celdas y la acción de limpiar; sus Sensors son la percepción de suciedad cercana, obstáculos, otros Roombas y estaciones. Implementa una máquina de estados, por lo que toma decisiones secuenciales orientadas a metas (SEARCH_DIRT, GO_TO_DIRT, CLEAN, GO_TO_CHARGER, RECHARGE). No es fijo, se desplaza en la cuadrícula y actualiza su trayectoria usando BFS evitando obstáculos y otros Roombas.

2. **DirtyCell**

El agente DirtyCell es un agente fijo, representando una celda con suciedad que debe ser limpiada. Su objetivo dentro del PEAS es pasivo: su Performance se mide por ser detectada y eliminada por el Roomba; su Environment es el grid donde coexiste con obstáculos y agentes móviles; no posee Actuators, y sus Sensors son nulos porque no percibe ni actúa. Es completamente reactivo, ya que simplemente espera a que un agente móvil llegue a su posición y lo remueva. No toma decisiones, no se mueve y su única función es proporcionar un objetivo dentro del entorno para que el RandomAgent lo detecte y lo limpie.

3. **ObstacleAgent**

El agente ObstacleAgent es un agente fijo que representa una barrera física dentro del grid. En términos de PEAS, su Performance se resume en mantener su posición y evitar que los agentes móviles lo atraviesen; el Environment es el grid que comparte con los otros agentes; no tiene Actuators ni Sensors, ya que no realiza acciones ni percibe nada. Es completamente reactivo y estático, actuando únicamente como una restricción que los Roombas deben evitar durante la planificación del camino. Aunque no participa activamente en la simulación, modifica el comportamiento del RandomAgent al influir en su búsqueda BFS y en su decisión para elegir caminos libres.

4. **ChargingStation**

El agente ChargingStation es un agente fijo cuya función es permitir que los RandomAgent recarguen su batería. Su objetivo dentro del PEAS se basa en servir como punto seguro de energía: su Performance puede verse como la capacidad de mantener operativos a los agentes móviles; su Environment es la misma cuadrícula; no posee Actuators, pero sí funciona como un Sensor pasivo al permitir que los agentes móviles detecten que están sobre ella. Es un agente reactivo, pues no actúa por sí mismo ni toma decisiones, solo proporciona el estado cargando al RandomAgent. También cumple un rol informativo: cuando un Roomba la detecta, se registra en la lista de estaciones conocidas del modelo, lo que permite colaboración indirecta entre agentes (simulación 2 solamente).

d) Arquitectura de subsunción de los agentes

El Roomba utiliza una arquitectura de subsunción donde sus comportamientos están organizados en capas jerárquicas que se activan dependiendo del estado del entorno y del propio agente. La máquina de estados de la simulación 1 y simulación 2 varían en cuanto a complejidad, pero la idea es la misma: dependiendo de su entorno y de su propio estado, el roomba elige si debe recargar batería, moverse o limpiar.

La capa más baja es la de movimiento básico, encargada de avanzar siguiendo rutas generadas por BFS. Esto evita colisiones y obstáculos. Encima de ella se ubica la capa de detección y limpieza, que toma control cuando el agente entra a una celda sucia y ejecuta la

acción de limpiar inhibiendo cualquier otro comportamiento. La siguiente capa es la de búsqueda de suciedad, que se activa cuando no hay suciedad cercana y obliga al agente a calcular el camino hacia la celda sucia más cercana. Por encima se encuentra la capa de gestión de batería, que se superpone a todas las anteriores: cuando la batería baja de un umbral seguro, esta capa ayuda a que el agente haga caso omiso del resto tanto la búsqueda como la limpieza, forzando al agente a dirigirse inmediatamente a la estación de carga siguiendo el camino calculado. Finalmente, en el nivel más alto está la capa de recarga, que mantiene al Roomba detenido en su estación hasta recuperar suficiente energía y liberar el control. Esta estructura garantiza que siempre se prioricen las necesidades más críticas (energía y supervivencia) sobre tareas funcionales cómo limpiar o desplazarse.

e) Características del ambiente

El ambiente de la Simulación 1 es inaccesible, porque el Roomba solo percibe su celda actual y las vecinas inmediatas sin conocer el estado completo del grid. Es no determinista, ya que la colocación aleatoria de suciedad y obstáculos y el consumo de batería producen comportamientos y trayectorias no completamente predecibles. También es no episódico, pues cada acción afecta estados futuros: limpiar reduce batería, moverse cambia el camino disponible y el agente debe recordar su ruta o estado interno. El entorno es dinámico, debido a que cambia continuamente cuando el agente limpia, pasando celdas sucias a limpias. Finalmente, es discreto, tanto en espacio como en tiempo, ya que opera sobre una cuadrícula de celdas finitas y avanza en pasos discretos hasta un máximo de 200 ciclos.

El ambiente de la Simulación 2 es inaccesible, porque cada Roomba solo percibe su celda actual y las celdas vecinas, sin acceso al mapa completo ni a la ubicación de todas las estaciones excepto las que descubre durante la simulación. Es no determinista, ya que los agentes pueden bloquearse entre sí, evitarse, y la distribución inicial aleatoria de suciedad y obstáculos produce rutas impredecibles, además de que las decisiones dependen de la disponibilidad dinámica de caminos. No es episódico, pues las acciones tienen consecuencias futuras: limpiar reduce batería, moverse cambia distancias a estaciones, descubrir una estación modifica el conocimiento compartido entre agentes y altera futuras decisiones. El entorno es dinámico, debido a que cambia constantemente por el movimiento simultáneo de múltiples Roombas, la eliminación de suciedad y la actualización del conjunto de estaciones conocidas. Finalmente, es discreto, ya que ocurre sobre una cuadrícula finita donde los cambios suceden en pasos de tiempo definidos y cada movimiento, limpieza o recarga sucede de acuerdo a ese tiempo máximo de ejecución.

En la Simulación 2, los Roombas también operan bajo una arquitectura de subsunción, organizada en capas de control donde las superiores inhiben a las inferiores según las demandas del entorno. La capa más básica es la de evitación de colisiones, encargada de impedir movimientos hacia celdas ocupadas por obstáculos u otros Roombas, lo cual garantiza seguridad y previene bloqueos. Encima se encuentra la capa de navegación, que permite al agente desplazarse por rutas generadas con BFS hacia su objetivo actual, ya sea una celda sucia o una estación. Sobre esta capa está la de detección y limpieza, que toma el control cuando el agente encuentra suciedad en su celda y ejecuta la acción de limpiar sin esperar a otros procesos. La siguiente capa es la de búsqueda de suciedad, que se activa cuando no hay suciedad visible y obliga al Roomba a calcular el camino hacia la celda sucia más cercana evitando otros robots. La capa superior es la de gestión de energía, la cual tiene prioridad sobre el resto: si la batería es insuficiente para regresar a una estación conocida, esta capa suprime toda búsqueda o limpieza y obliga al agente a dirigirse a cargar.

Finalmente, la capa más alta es la de aprendizaje y compartición de estaciones, que recopila estaciones nuevas al visitarlas y actualiza la "memoria" del modelo. Este orden jerárquico asegura que los agentes reaccionen primero a los factores de alta importancia (colisiones y batería), luego a las tareas usuales (limpiar y navegar) y utilizar conocimiento adquirido para mejorar su desempeño.

Análisis de Resultados

a) Máquinas de estados

Las máquinas de estados de ambas simulaciones comparten la estructura básica de buscar suciedad, moverse hacia ella, limpiarla y gestionar la batería, pero difieren en complejidad y prioridades. En la Simulación 1, la máquina de estados es explícita y estructurada en cinco estados bien definidos (SEARCH_DIRT, GO_TO_DIRT, CLEAN, GO_TO_CHARGER, RECHARGE), lo que permite una lógica clara basada en percepción local sin interacción con otros agentes. Por otro lado, en la Simulación 2, la máquina de estados es más simple y reactiva, reducida a los estados principales search_dirty y go_charge, pero integrada dentro de una arquitectura de subsunción donde la evitación de colisiones, la búsqueda de estaciones conocidas y el movimiento aceptable según las condiciones tienen prioridad sobre la lógica básica. Esto hace que la Simulación 2 no requiera tantos estados explícitos, porque capas superiores (como evitar otros Roombas y compartir estaciones) suprimen o modifican el flujo de control. En resumen, la Simulación 1 tiene una máquina de estados más detallada, mientras que la Simulación 2 usa una máquina de estados más compacta, apoyada en una jerarquía de comportamientos que toma decisiones dinámicas según el entorno y la presencia de otros agentes.

b) Desempeño

En la simulación del Roomba, se observa que el número de agentes influye directamente en la eficiencia de limpieza. Cuando se incrementa la cantidad de Roombas, el cuarto se limpia significativamente más rápido; por ejemplo, con 5 Roombas, el porcentaje de celdas limpias alcanza niveles altos entre 50 y 70 pasos, mientras que con un solo Roomba el tiempo necesario se duplica o incluso puede superar el tiempo máximo de ejecución definido, dejando celdas sin limpiar. Esto demuestra que la paralelización del trabajo permite cubrir más área del grid simultáneamente.

c) Observaciones

Otro factor importante es la distribución de las estaciones de carga. Tener varias estaciones evita conflictos entre los Roombas, ya que cada agente puede recargar sin interferir con los demás, evitando que se "peleen" o bloqueen por una estación única. Esto también reduce el tiempo muerto de los Roombas esperando para recargar y aumenta la eficiencia general del sistema.

Una observación adicional son los cambios que se le hicieron a la simulación para que funcionara de mejor manera, cambios que fueron únicamente observados y necesarios en la segunda simulación. Como lo son la necesidad de compartir información de estaciones de carga y áreas ya limpiadas por los roombas cuando estos son vecinos (que igualmente podría ser una lista a la que tengan acceso, pero quizás no sea lo más realista).

Conclusiones

La simulación demuestra que la arquitectura de subsunción es altamente efectiva para controlar agentes móviles en un entorno dinámico, permitiendo que los Roombas prioricen tareas críticas como la gestión de energía y la evitación de colisiones mientras continúan cumpliendo su objetivo principal de limpieza. La comparación entre la versión de un agente y el sistema multiagente evidencia que la colaboración, la distribución espacial del trabajo y el intercambio de información mejoran significativamente la eficiencia total del sistema. Adicionalmente, se observa que características como la ubicación de estaciones de carga, la densidad de suciedad y los obstáculos influyen directamente en el desempeño global. En conjunto, los resultados resaltan la importancia de diseñar agentes autónomos con capacidades reactivas, cooperativas y energéticamente conscientes para resolver tareas complejas de manera eficiente.

Anexos