Tecnológico de Monterrey

Act.

Luis Leonardo Rodriguez Galvez
a01029331

Prof. Octavio Navarro Hinojosa

Modelación de sistemas multiagentes con gráficas computacionales

Gpo. 302

19 de Noviembre del 2025

1. Definición del Problema y Propuesta de Solución

El Problema: El reto consiste en limpiar un espacio desconocido de dimensiones finitas (una cuadrícula) que contiene suciedad distribuida aleatoriamente y obstáculos que impiden el paso libre. El problema principal no es solo "limpiar", sino hacerlo de manera autónoma y eficiente, asegurando que el agente (robot) no se quede sin energía en medio de la tarea. Un movimiento puramente aleatorio sería ineficiente y peligroso para la viabilidad del agente a largo plazo.

La Solución Propuesta: Implementé una simulación basada en agentes utilizando la librería mesa. Mi propuesta se centra en un agente "inteligente" (aunque la clase se llame RandomAgent, su comportamiento es cognitivo y reactivo) que construye un mapa interno de su entorno a medida que lo explora. La solución integra algoritmos de búsqueda (BFS) para la navegación y una **arquitectura de subsunción** para la toma de decisiones, priorizando siempre la supervivencia (energía) sobre la misión (limpieza) y la exploración.

2. Diseño de los Agentes

Para este modelo, diseñe cuatro tipos de agentes, siendo el RandomAgent el protagonista y los otros tres elementos pasivos del entorno.

Agente Principal: Robot de Limpieza (RandomAgent)

- **Objetivo:** Maximizar la cantidad de celdas limpias (recolectar TrashAgent) mientras gestiona su nivel de batería para no "morir" lejos de una estación de carga.
- **Capacidad Efectora (Acciones):**
    - **Moverse:** Desplazarse a una de las 8 celdas vecinas (Vecindad de Moore).
    - **Limpiar:** Recoger basura en su propia celda.
    - **Cargar:** Recuperar energía al estar sobre una ChargingStationAgent.
    - **Escanear:** Percibir el entorno inmediato y actualizar su mapa interno.
- **Percepción:**
    - El agente "ve" su vecindad inmediata (vecinos ortogonales y diagonales).
    - Identifica obstáculos, basura, estaciones de carga y terreno vacío.
    - **Memoria:** Mantiene un diccionario (self.mapa) donde registra qué celdas ha visitado (2), cuáles son obstáculos (1), cuáles son libres (0) y cuáles son "frontera" o desconocidas (-1).
- **Métricas de Desempeño Individual:**
    - trash_count: Cantidad de basura recogida.
    - movement_count: Cantidad de pasos dados (eficiencia de movimiento).
    - energy: Nivel de batería restante.

Agentes de Entorno (FixedAgents)

- **TrashAgent:** Representa la suciedad. Desaparece cuando el robot interactúa con ella (remove()).
- **ObstacleAgent:** Representa paredes o muebles. Bloquea el movimiento y la visión de ruta del robot.
- **ChargingStationAgent:** Puntos fijos donde el robot regenera energía.

3. Arquitectura de Subsunción (Jerarquía de Acciones)

En mi método move(), implementé una arquitectura de subsunción estricta. El agente no evalúa "qué es mejor en promedio", sino que sigue una jerarquía de inhibición: si una condición de mayor prioridad se cumple, inhibe a las de menor prioridad.

Jerarquía de Comportamientos (de mayor a menor prioridad):

1. **Supervivencia Inmediata (Gestión de Carga):**
    - Condición: ¿Estoy cargando (state_charging)?
    - Acción: Si la energía no está al 100%, me quedo quieto y cargo. Si está llena, me desconecto y paso al siguiente nivel.
2. **Supervivencia Preventiva (Batería Crítica):**
    - Condición: ¿Mi energía es menor a 40 unidades?
    - Acción: Ignoro la basura y la exploración. Calculo la ruta más corta (BFS) a la estación conocida más cercana (go_to_station) y voy hacia ella.
3. **Misión Principal (Recolección Oportunista):**
    - Condición: ¿Veo basura en alguna celda vecina accesible?
    - Acción: Me muevo a esa celda inmediatamente (en el siguiente step la limpiaré).
4. **Exploración Local (Frontera Inmediata):**
    - Condición: ¿Tengo un vecino que nunca he visto ni visitado (marcado como -1 en el mapa)?
    - Acción: Me muevo a esa celda desconocida para expandir el mapa rápidamente.
5. **Exploración Proactiva (Planificación Global):**
    - Condición: No hay nada interesante cerca.
    - Acción: Busco en mi memoria (self.mapa) la celda "desconocida" (-1) más cercana a mi posición actual (distancia Manhattan) y trazo una ruta segura usando BFS hasta ella.

4. Características del Ambiente (RandomModel)

El entorno simula una habitación interior con las siguientes propiedades:

- **Espacio Discreto:** Una cuadrícula (OrthogonalMooreGrid) de tamaño $W \times H$.
- **Límites Físicos:** Implementé un borde de obstáculos alrededor de toda la grilla para simular las paredes de la habitación.
- **Obstáculos Aleatorios:** Además de las paredes, un porcentaje (porObs, por defecto 20%) de las celdas interiores son obstáculos, creando laberintos y pasillos.
- **Suciedad Estocástica:** La basura aparece con una probabilidad (probTrash) en las celdas vacías.
- **Dinámica de Energía:** Cada movimiento cuesta 1 unidad de energía. Limpiar cuesta 1 unidad adicional. La carga recupera 5 unidades por turno.

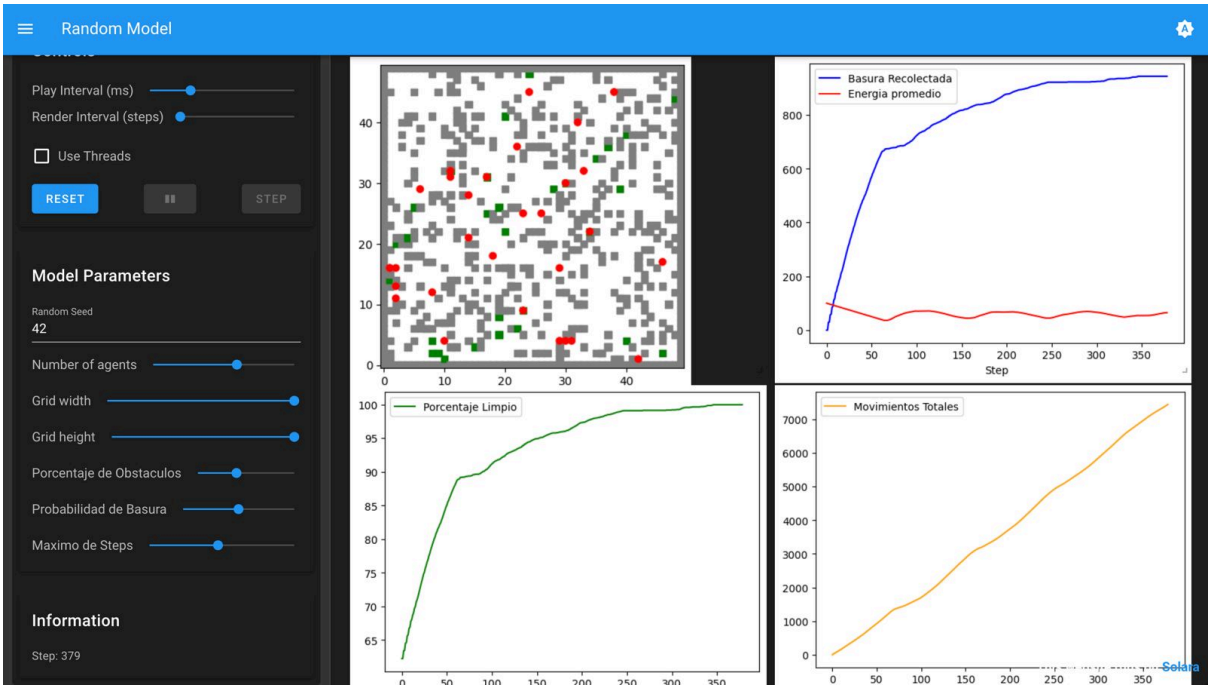
5. Estadísticas Recolectadas

Para evaluar el éxito de mi diseño, configuré el DataCollector para extraer la siguiente información paso a paso:

1. **Basura Recolectada:** Mide la eficacia directa del cumplimiento de la misión.
2. **Energía Promedio:** Me permite saber si los agentes están gestionando bien sus recursos o si están al borde de la "muerte".
3. **Porcentaje de Celdas Limpias:** Una métrica global del estado del ambiente.

4. **Movimientos Totales:** Ayuda a calcular la eficiencia (ej. basura recolectada por paso dado).

6. Análisis de Resultados (Simulación 30 Agentes)



PEAS (Performance & Environment)

Performance (Medida de Rendimiento): Mi medida principal de éxito fue alcanzar el 100% de limpieza en el menor número de pasos posible manteniendo a los agentes vivos.

- **Resultado Cuantitativo:** Se logró limpiar el entorno completo en **379 pasos**.
- **Eficiencia Energética:** La gráfica de "Energía promedio" (línea roja) se mantiene oscilando establemente entre 40 y 80 unidades, lo que indica que ningún agente murió y el ciclo de recarga (prioridad 1 y 2 en mi arquitectura) funcionó correctamente para mantener la flota operativa.
- **Tasa de Limpieza:** La gráfica de "Basura Recolectada" muestra un crecimiento logarítmico: muy rápido al inicio y lento al final.

Environment (Entorno):

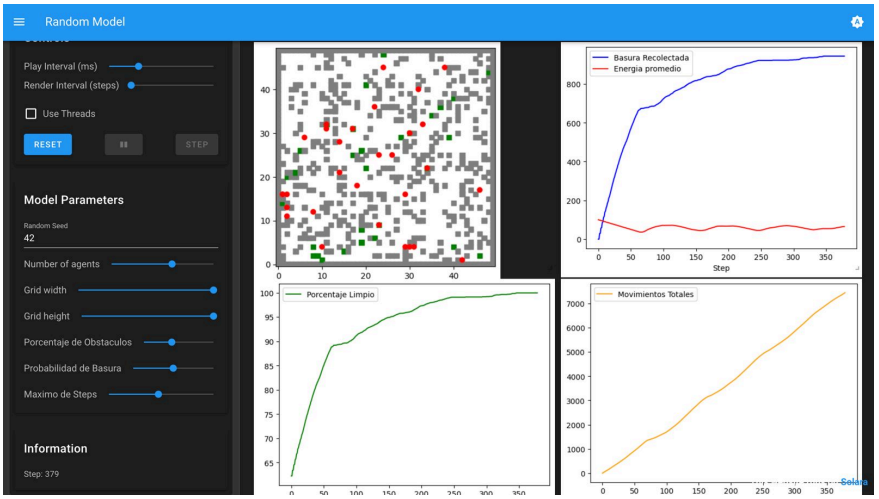
- **Espacio:** Grid de 50x50 (2,500 celdas totales).
- **Complejidad Estática:** 20% de obstáculos, lo que crea un laberinto moderado que fuerza el recálculo de rutas BFS.
- **Carga de Trabajo:** 50% de suciedad inicial (alta densidad).
- **Dinámica Social:** 30 agentes compitiendo por espacio. Al haber una densidad de agentes del 1.2%, es probable que ocurrieran bloqueos momentáneos (wait) al intentar acceder a pasillos estrechos o estaciones de carga.

Conclusiones de esta Ejecución

Basado en la gráfica de "Porcentaje Limpio" (abajo izquierda), observó un comportamiento clásico de **rendimientos decrecientes**. Los agentes limpiaron el primer 90% de la basura muy rápido (en los primeros aprox 100 pasos) debido a la alta probabilidad de encontrar suciedad por azar o vecindad. Sin embargo, el último 10% tomó casi el triple de tiempo (pasos 100 a 379).

Esto valida mi implementación de la **Exploración Proactiva**: cuando la basura deja de ser obvia, los agentes tuvieron que usar su memoria (self.mapa) para viajar a zonas "no visitadas" lejanas, lo que explica por qué la limpieza continuó hasta el final en lugar de estancarse. El sistema es robusto: 30 agentes lograron coordinarse indirectamente para cubrir un mapa grande sin agotarse.

Análisis de Resultados (Simulación 10 Agentes)



PEAS (Performance & Environment)

Performance (Medida de Rendimiento):

- **Tiempo de Ejecución:** La limpieza total se completó en **198 pasos**. Al ser un mapa más pequeño, la convergencia fue mucho más rápida.
- **Comportamiento Energético Crítico:** A diferencia de la simulación anterior, aquí la gráfica de "Energía promedio" (línea roja) muestra una caída dramática cerca del paso 70, bajando hasta casi 35 unidades antes de recuperarse. Esto indica una **sincronización de fatiga**: la mayoría de los agentes agotaron sus baterías casi al mismo tiempo y tuvieron que activar su "Instinto de Supervivencia" (ir a cargar) simultáneamente.
- **Eficacia:** Se logró el 100% de limpieza (gráfica verde), validando que incluso con fluctuaciones de energía, la misión se cumple.

Environment (Entorno):

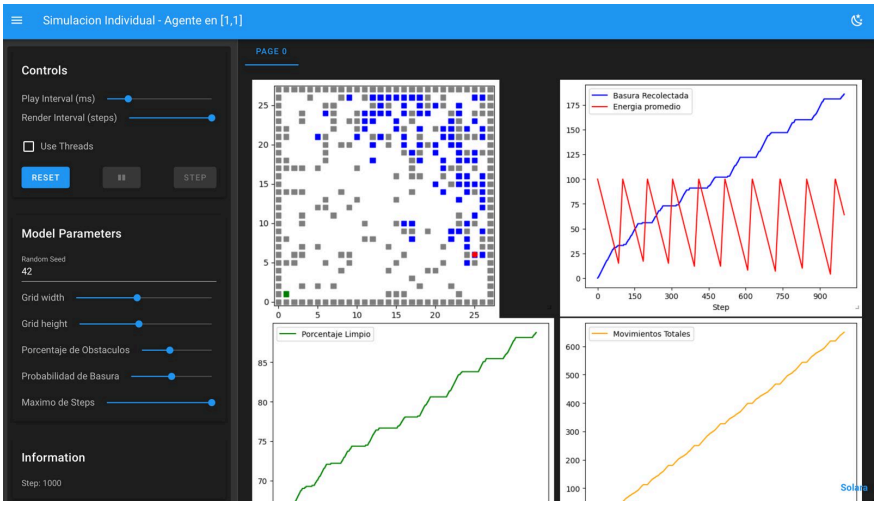
- **Espacio:** Grid de 25x25 (625 celdas). Es un cuarto del tamaño del mapa anterior.
- **Densidad:** 10 agentes resultan en una densidad del 1.6%, ligeramente superior a la prueba anterior.
- **Obstáculos y Basura:** Se mantuvieron las proporciones (20% obstáculos, 50% basura), lo que garantiza que la dificultad topológica es equivalente, solo que a menor escala.

Conclusiones de esta Ejecución

Lo más destacable de esta prueba es la **resiliencia del sistema ante la crisis energética**. En el paso 70, la flota estuvo peligrosamente cerca del umbral crítico. Si mi arquitectura de subsunción no tuviera la prioridad de "Ir a cargar" por encima de "Limpiar", los agentes habrían muerto en ese valle.

La recuperación posterior de la línea roja demuestra que el algoritmo de búsqueda (BFS) hacia las estaciones funcionó perfectamente. Además, al ser menos agentes, el "ruido" estadístico es menor y se notan más los ciclos de trabajo/descarga (como un latido) en comparación con la gráfica más plana de los 30 agentes.

Análisis de Resultados (Simulación Individual - 1 Agente)



PEAS (Performance & Environment)

Performance (Medida de Rendimiento):

- **Patrón de "Dientes de Sierra":** La gráfica de energía (línea roja) es el resultado más fascinante de esta prueba. Muestra oscilaciones perfectas y constantes. El agente consume energía linealmente hasta llegar al umbral crítico (~40%), regresa a la base, se carga al 100% y vuelve a salir. Esto confirma que la lógica de supervivencia funciona impecablemente en aislamiento.
- **Efecto "Escalera" en la Limpieza:** A diferencia de los modelos multi-agente donde la limpieza es continua, aquí la gráfica de "Basura Recolectada" y "Porcentaje Limpio" tiene forma de escalera. Las mesetas (líneas horizontales) representan el **tiempo muerto** donde el agente está viajando a la base o cargando.
- **Lentitud Inherente:** Al paso 1000, el agente aún no ha terminado (lleva aprox. 88%). Esto demuestra que, aunque el agente es autónomo y capaz, la falta de paralelismo hace que la tarea sea exponencialmente más lenta.

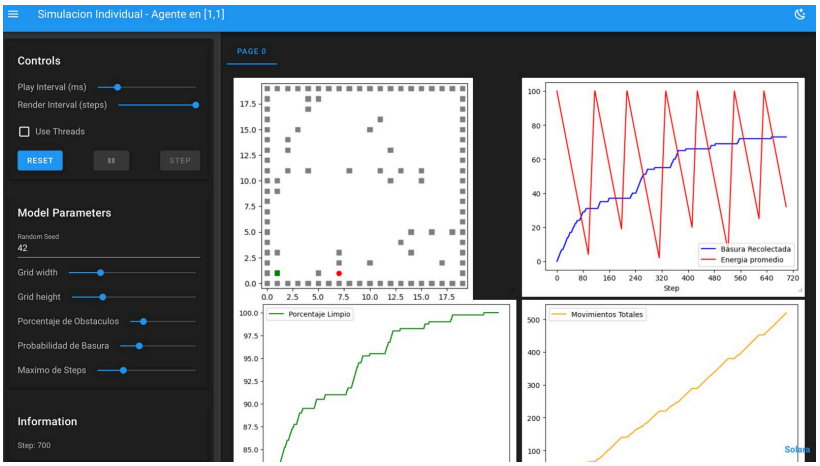
Environment (Entorno):

- **Soledad:** Al no haber otros agentes, no hay conflictos de espacio ni colas en la estación de carga. El agente tiene libertad total de movimiento.
- **La Tiranía de la Distancia:** A medida que el agente limpia las zonas cercanas a la base, debe viajar más lejos para encontrar suciedad nueva. Esto hace que cada "ciclo de trabajo" sea menos productivo, ya que gasta más batería en el trayecto (commuting) que en la limpieza efectiva.

Conclusiones de esta Ejecución

Esta prueba valida la robustez del código: el agente es capaz de sobrevivir indefinidamente (ciclos infinitos de carga y descarga) sin quedarse varado. Sin embargo, ilustra la limitación física de un solo efector. La eficiencia cae drásticamente conforme la frontera de exploración se aleja de la estación de carga. Es la prueba definitiva de por qué necesitamos sistemas multi-agente: no solo para limpiar más rápido, sino para distribuir geográficamente el gasto energético.

Análisis de Resultados (Simulación Individual - Grid Pequeño 20x20)



PEAS (Performance & Environment)

Performance (Medida de Rendimiento):

- **Eficacia de Terminación:** A diferencia de la prueba anterior (25x25) donde el agente no terminaba ni en 1000 pasos, aquí logró limpiar el 100% en **700 pasos**. Esto resalta cómo la dificultad para un solo agente no crece linealmente con el tamaño del mapa, sino de forma mucho más agresiva debido a los tiempos de desplazamiento.
- **Ciclos de Carga Críticos:** La gráfica de energía (línea roja) muestra un patrón de "dientes de sierra" muy agresivo. Hubo momentos (cerca del paso 80 y 320) donde la energía tocó fondo, casi llegando a 0. Esto indica que mi umbral de retorno (40 unidades) es funcional pero arriesgado en mapas con obstáculos complejos, ya que el camino de vuelta consumió casi toda la reserva.

- **Productividad Intermitente:** La gráfica de "Basura Recolectada" (azul) confirma el efecto "escalera". Las mesetas horizontales son largas, lo que significa que el agente pasó una gran parte de esos 700 pasos simplemente viajando (`commuting`) entre la estación y la zona de trabajo, sin limpiar nada.

Environment (Entorno):

- **Espacio Reducido:** Grid de 20x20 (400 celdas). Al ser más pequeño, la "frontera desconocida" está, en promedio, más cerca de la estación que en el mapa de 25x25.
- **Densidad de Obstáculos:** Mantenemos el 20%. En un mapa chico, esto puede crear "islas" o pasillos ciegos que obligan al agente a gastar mucha energía retrocediendo.

Conclusiones de esta Ejecución

Esta prueba cierra el ciclo de validación demostrando los límites físicos del agente individual.

1. **La Tiranía del Desplazamiento:** Aunque el agente es capaz de limpiar todo, se vuelve ineficiente con el tiempo. Al principio limpia rápido (gráfica verde empinada), pero al final (pasos 400-700) tarda mucho más porque la basura restante está en las esquinas más lejanas a su base.
2. **Validación de Supervivencia:** El agente sobrevivió a 7 ciclos de carga/descarga completos. Esto valida que el algoritmo de búsqueda de camino (BFS) hacia la estación es robusto incluso cuando la batería es crítica.
3. **Comparativa de Escala:** Reducir el mapa de 25x25 a 20x20 (una reducción del 36% en área) permitió que el agente terminara el trabajo, bajando el tiempo de >1000 pasos a 700.

CONCLUSIONES:

¿Por qué es eficiente y por qué no?

- **ES EFICIENTE PORQUE:** Logra el objetivo completo (limpiar todo) sin intervención humana y gestiona sus recursos autónomamente. La implementación de estados (Charging, Cleaning, Exploring) evita comportamientos erráticos.
- **NO ES EFICIENTE PORQUE:** Carece de **comunicación directa**. Los agentes no se dicen "yo voy a limpiar la zona A, tú ve a la B". Esto provoca que a veces dos agentes vayan a la misma basura, o que sigan explorando el mismo rincón, desperdiciando movimientos (overlap).

Conclusión Final

El diseño del RandomAgent (que irónicamente es muy poco "random" y bastante cognitivo) demuestra que dotar a los agentes de memoria espacial y un instinto de supervivencia es la clave para resolver tareas de cobertura completa. El modelo validó que la cantidad vence a la calidad individual: un enjambre de robots simples supera por mucho en tiempo y eficiencia energética a un solo robot sofisticado que debe viajar largas distancias.