Tecnológico de Monterrey

Sistema de Agentes de Limpieza (Roomba)

Juan Pablo Narchi A01781518

18 de noviembre de 2025

Modelación de sistemas multiagentes con gráficas computacionales

Octavio Navarro

Introducción

Los sistemas multiagentes representan o forman parte de lo que conocemos hoy como inteligencia artificial donde múltiples sistemas colaboran para resolver problemas o tareas complejas. En este proyecto simulamos aspiradoras robóticas (Roomba) que limpian espacios de forma autónoma, gestionando batería limitada, evitando obstáculos y descubriendo estaciones de carga en su entorno. Se implementan dos simulaciones usando la librería Mesa: una con un solo agente y otra con múltiples agentes de limpieza, aplicando exploración inteligente con memoria de celdas visitadas para optimizar la cobertura del espacio.

Problema y Propuesta de Solución

Se desarrolla un sistema de agentes para limpiar celdas (o habitaciones) de forma eficiente, considerando limitaciones de batería, obstáculos, con celdas distribuidas aleatoriamente. Los agentes reactivos usan un algoritmo de exploración inteligente con memoria de celdas visitadas para maximizar la cobertura del espacio. Se realizan dos simulaciones: la primera con un agente individual y estación de carga en (0,0), y la segunda con múltiples agentes colaborativos con estaciones de carga individuales. El sistema utiliza la librería Mesa para la simulación.

Diseño de los Agentes

RandomAgent (Agente de Limpieza)

Objetivo: Limpiar el máximo de celdas sucias, gestionando la batería eficientemente.

Capacidad Efectora: Moverse en 8 direcciones (vecindad de Moore), limpiar celdas sucias, cargar batería en estaciones y buscar rutas óptimas hacia celdas u estaciones.

Percepción*:* Conoce su nivel de batería, sus vecinos inmediatos (8 celdas), ubicación de todas las celdas sucias y estaciones de carga (listas globales), y su estado de carga.

Reactividad: Es un agente reactivo que explora priorizando celdas no visitadas, descubre estaciones de carga en su vecindario, anticipa recargas cuando batería < 40%, prioriza tareas y toma decisiones autónomas sin intervención externa.

Arquitectura de Subsunción

La arquitectura define una jerarquía de comportamientos donde las capas superiores tienen prioridad sobre las inferiores:

Jerarquía de Comportamientos:

1. **Supervivencia (Máxima):** Si batería ≤ 0% = Agente muere (is_dead = True)
2. **Gestión de Carga***:* Si batería < 100% y está cargando = Continuar carga (+5% por step); si batería ≥ 100% = Desactivar modo carga
3. **Búsqueda de Carga***:* Si batería < 40% y no está cargando = Ir a estación más cercana conocida y activar modo carga
4. **Limpieza Inmediata***:*Si hay celda sucia en vecindad = Limpiar, moverse y eliminar del registro (consumo: 1%)
5. **Exploración Inteligente (Mínima)***:* Si no hay celdas sucias = Terminar; sino explorar priorizando celdas no visitadas (consumo: 1%)

Algoritmo de Exploración Inteligente: El agente mantiene una memoria de celdas visitadas (visited_cells). Al explorar, prioriza moverse a celdas no visitadas de su vecindario. Si todas las celdas vecinas ya fueron visitadas, elige aleatoriamente entre ellas.

Algoritmo Nearest Neighbor Search para encontrar Estación de Carga: Calcula la distancia euclidiana para encontrar el objetivo más cercano usando la fórmula: $distancia = \sqrt{[(x\_cel - x\_objetivo)^2 + (y\_cel - y\_objetivo)^2]}$. Se aplica para encontrar las estaciones de carga cercanas pasando todos los neighbors al algoritmo para identificar de los 8 vecinos cuál es la celda más cercana.

Características del Ambiente

Característica	Valor
Tipo de Grid	OrthogonalMooreGrid (8 vecinos)
Dimensiones	28 × 28 celdas
Topología	No toroidal (bordes cerrados)
Tipo de ambiente	Observable, Determinístico, Episódico, Estático, Discreto

Agentes en el ambiente

RandomAgent (Agente de Limpieza): Color rojo (negro con X si está muerto), forma circular, batería inicial 100%, recarga 5% por step en estación, consumo por movimiento 1%, consumo por limpieza 1%.

DirtyAgent (Celda Sucia): Color verde, forma estrella (*), cantidad de 10 celdas por defecto, distribución aleatoria, comportamiento estático.

ChargingStation (Estación de Carga): Color azul, forma cuadrada. Sim1: coordenada (0,0). Sim2: posiciones aleatorias (una por cada agente). Comportamiento estático.

ObstacleAgent (Obstáculo): Color gris, forma cuadrada, cantidad 10 obstáculos inicialmente (se puede modificar los parámetros), distribución aleatoria, comportamiento fijo (no se puede atravesar).

Estadísticas de Simulaciones

Configuración Simulación 2:

- **Número de agentes:** 10 (configurable de 1 a 50)
- **Grid:** 28 × 28 (784 celdas totales)
- **Celdas sucias:** 10 (configurable de 1 a 200)
- **Obstáculos:** 10 (configurable de 1 a 200)
- **Posiciones iniciales:** Aleatorias
- **Consumo por limpieza:** 1% de batería

- **Consumo por movimiento:** 1% de batería
- **Estaciones de carga:** Una por cada agente, ubicada en su posición inicial

Características Especiales

- Cada agente tiene su propia estación de carga
- Los agentes descubren estaciones de otros agentes durante la exploración
- Múltiples agentes colaboran para limpiar el espacio

Configuración Simulación 1:

- **Número de agentes:** 1
- **Grid:** 28 × 28 (784 celdas totales)
- **Celdas sucias:** 10 (configurable de 1 a 50)
- **Obstáculos:** 10 (configurable de 1 a 50)
- **Posición inicial del agente:** (1,1)
- **Consumo por limpieza:** 1% de batería
- **Consumo por movimiento:** 1% de batería
- **Estación de carga:** Ubicada en coordenada (0,0)

Características Especiales

- Un solo agente explora todo el espacio
- El agente conoce la ubicación de la estación de carga desde el inicio
- Usa memoria de celdas visitadas para exploración eficiente

Métricas de Visualización

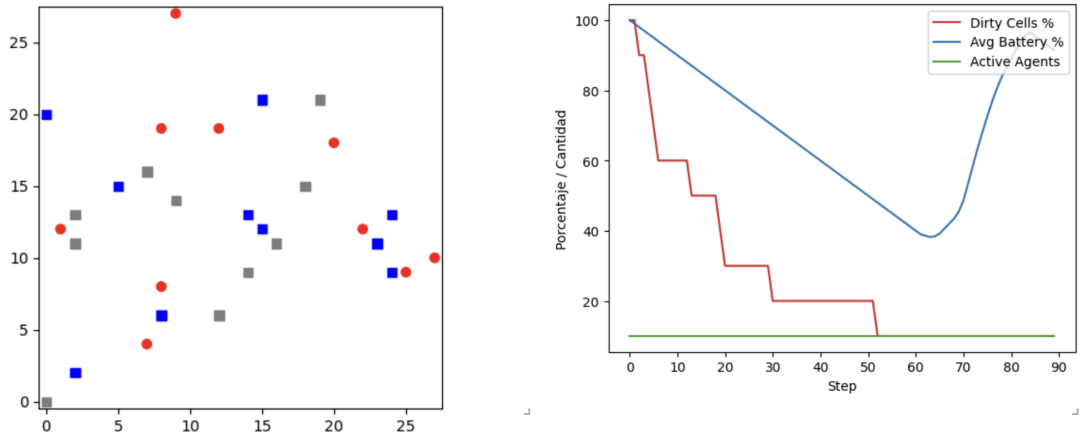
Ambas simulaciones incluyen un gráfico en tiempo real que muestra:

- Dirty Cells % (rojo): Porcentaje de celdas sucias restantes
- Avg Battery % (azul): Batería promedio de los agentes activos
- Active Agents (verde): Número de agentes que siguen vivos

Resultado Simulación 2

Lo que observamos en el gráfico es que después de 90 steps, la recolección de las celdas sucias se completa usando solamente 10 agentes. Calculando el promedio de las 5 simulaciones que corrimos: 90, 360, 109, 212 y 53 nos da un promedio de 174.8 steps. Pero lo interesante es que si agregamos un agente más a la simulación, es decir, comenzamos la simulación con 11 agentes, conseguimos estas nuevas métricas con un promedio de steps de 118. Es decir que si agregamos un solo agente más cada vez, aproximadamente vamos a reducir el tiempo de steps en un 47%.

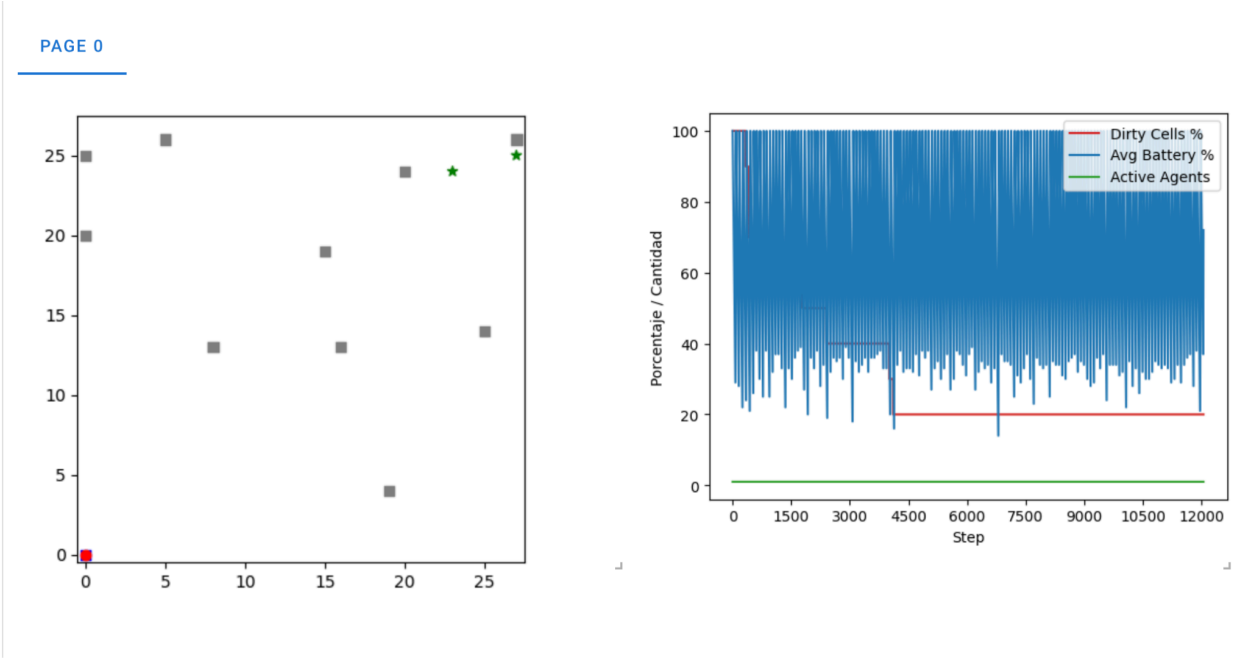
PAGE 0



Resultado Simulación 1

A diferencia de la simulación 2, en la simulación 1, donde nada más hay un solo agente, le cuesta mucho trabajo terminar de concluir la limpieza de todas las celdas. Es decir, que tuve resultados donde se alargó a más de 10.000 steps para poder limpiar todas las celdas sucias y todavía no terminaba. Esto porque, al no tener ayuda de otros agentes, tiene que desplazarse por todo el ambiente el agente individual y asegurarse de que todas las celdas estén limpias.Se crean muchos steps, porque al mismo tiempo que la gente explora su entorno de una forma aleatoria, un poco

inteligente, siempre tiene que regresar a la base, siempre y cuando la batería sea menor a 40. Esto también provoca que las barras se muestran como en el gráfico de a continuación:



Podemos ver en la simulación que llevábamos 12,000 steps y aún restaban dos celdas sucias. Y es decir que estas dos celdas sucias estaban muy lejos de la celda de la gente, por lo cual era muy complicado llegar hasta ellas con el algoritmo que implementamos de exploración.

Conclusión

Me llamo mucha la atención la tarea y le heche muchas ganas a entender a profundidad y hacer un sistema inteligente. Esta creo que es mi primera introducción hacia sistemas “inteligentes” por sí mismo y que imitan un razonamiento de algo que piensa. Disfruto mucho el programar esto y me ayuda mucho a crear entornos de agentes más complejos en el futuro.

Referencias

Comprender el análisis de distancia euclidiana—ArcGIS Pro | Documentación. (n.d.). https://pro.arcgis.com/es/pro-app/3.3/tool-reference/spatial-analyst/understanding-euclidean-distance-analysis.htm