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 coordinando entre sí mediante algoritmos de búsqueda (Nearest Neighbor). Se implementan dos simulaciones usando la librería Mesa: una con un solo agente y otra con múltiples agentes de limpieza, aplicando optimización de rutas midiendo la distancia más corta en un plano euclidiano, con el cálculo de la distancia podemos saber cual es la mejor coordenada para que el agente vaya hacia la celda por limpiar, de rutas tipo Neighbor Search.

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 el algoritmo Nearest Neighbor Search para optimizar rutas de limpieza. Se realizan dos simulaciones: la primera con un agente individual y estación de carga fija, y la segunda con múltiples agentes colaborativos y una estación de carga. 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 calcula constantemente rutas cortas hacia objetivos, anticipa recargas cuando batería < 50%, prioriza tareas según subsunción 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% = Terminar simulación
2. **Gestión de Carga***:* Si batería < 100% y batería ≥ 100% = Continuar/desactivar carga según corresponda
3. **Búsqueda de Carga***:* Si batería < 50% y no está cargando = Ir a estación más cercana (Nearest Neighbor) y activar modo carga
4. **Limpieza Inmediata***:* Si hay celda sucia en vecindad = Limpiar, moverse y eliminar del registro (consumo: Sim1: 2%, Sim2: 1%)
5. **Exploración Inteligente (Mínima)***:* Si no hay celdas sucias = Terminar; sino calcular y moverse a celda más cercana (Nearest Neighbor, consumo: 1%)

Algoritmo Nearest Neighbor Search: 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 tanto celdas sucias como 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, forma circular, batería inicial 100%, recarga 5% por step en estación, consumo por movimiento 1%, consumo por limpieza Sim1: 2%, Sim2: 1%.

DirtyAgent (Celda Sucia): Color verde, forma cuadrada, cantidad 10 celdas, distribución aleatoria, comportamiento estático.

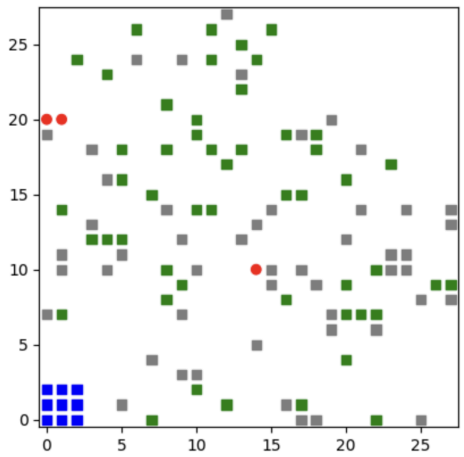
ChargingStation (Estación de Carga): Color azul, forma cuadrada. Sim1: esquina superior (alrededor de coordenada (1,1) con offset +25). Sim2: Esquina inferior izquierda + esquina (0,0). Comportamiento estático.

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

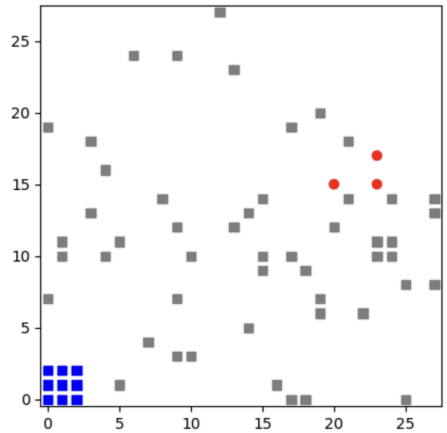
Estadísticas de Simulaciones

Configuración Simulación 2: Número de agentes: 3, Grid: 28 × 28 (784 celdas), Celdas sucias: 50,Obstaculos: 50, Posiciones iniciales: Aleatorias, Consumo por limpieza: 1% y por movimiento 1%, Una estación de carga abajo a la izquierda.

Inicio:



Final:

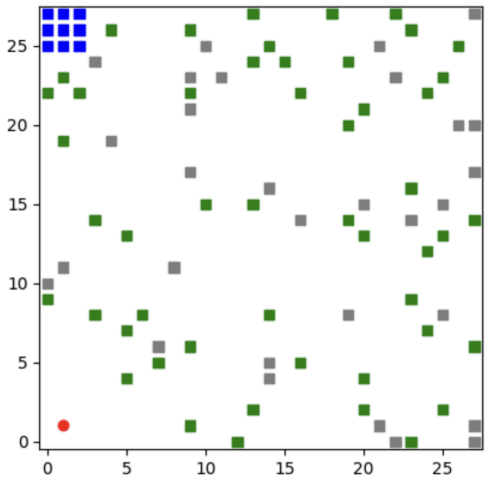


Resultado Simulación 2

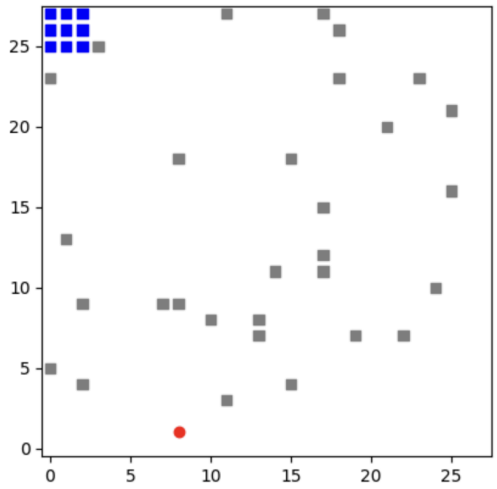
Hice 5 simulaciones con la misma configuración de entorno donde tuve estos resultados de steps totales cuando los agentes limpian todas las celdas: 99, 76, 107, 100 y 100. Esto nos indica que el promedio de steps para limpiar las 50 celdas son de 95.2 steps. Lo interesante es que si bajo los obstáculos a 27, los steps suben en un 36% en un promedio de 130, lo que creo que pasó es que al haber más obstáculos los agentes van a diferentes celdas sucias la mayoría de las veces pero con más obstáculos muchos se unen y van hacia la misma.

Configuración Simulación 1: Número de agentes: 1, Grid: 28 × 28 (784 celdas), Celdas sucias: 50,Obstaculos: 30, Posición inicial de agente: (1,1), Consumo por limpieza: 1% y por movimiento 1%, Una estación de carga arriba a la izquierda.

Inicio:



Final:



Resultado Simulación 2

Hice 4 simulaciones con la misma configuración de entorno donde tuve estos resultados de steps totales cuando los agentes limpian todas las celdas: 407, 327, 351 y 405. Esto nos indica que el promedio de steps para limpiar las 50 celdas son de 372 steps. Al bajar los obstáculos a 1 el agente tiene una mejora de 14% en rendimiento de limpieza respecto a los 30 obstáculos anteriores, esto hace sentido porque el agente no tiene tantas limitaciones al ir a la mejor ruta hacia la celda sucia.

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