



# Tecnológico de Monterrey

Act.

Héctor Lugo Gabino A01029811

Prof. Octavio Navarro Hinojosa

Modelación de sistemas multiagentes con gráficas computacionales

Gpo. 302

19 de Noviembre del 2025

## 1. Problema

El objetivo es modelar el comportamiento y analizar una simulación basada en agentes, implementando el comportamiento de robots aspiradores autónomos en un entorno discreto.

## 2. Descripción del Modelo

### 2.1 Ambiente

**Discreto:** El entorno se define como una cuadrícula de  $M \times N$ , donde los estados (sucio, limpio, obstáculo, vacío, estación) están claramente delimitados por celdas discretas.

**Observabilidad Parcial:** Los agentes no tienen acceso al estado global del grid. Solo pueden percibir su vecindad inmediata (Moore Neighborhood: las 8 celdas adyacentes) y construyen un mapa interno basado en lo que han explorado.

**Secuencial:** las decisiones actuales (moverse lejos) afectan drásticamente las posibilidades de quedarse sin batería antes de poder regresar.

**Dinámico:** Aunque el ambiente es estático, en tanto que los obstáculos no se mueven, cambia constantemente debido a las acciones de otros agentes, lo que cambia la percepción del ambiente de los agentes.

**No Determinista:** la distribución inicial de basura, obstáculos y la posición de los agentes es “aleatoria”. Además de que cuando el agente no tiene una meta clara (buscar caminos inexplorados, limpiar basura o regresar a su estación de carga), su movimiento de exploración es aleatorio.

**Semi-accesible:** el agente depende de sus sensores locales propios, y su memoria interna; no puede acceder a información de celdas no exploradas, a menos que las visite o vea alrededor de él.

### 2.2 Agente

**Autónomos:** operan sin la intervención humana, una vez iniciada la simulación deciden su acciones establecidas en el código, “tomando sus propias decisiones”.

**Reactividad:** reaccionan a estímulos inmediatos:

- Si perciben basura - entonces - limpian
- Si perciben batería baja - entonces - su estado cambia a “Regreso a la base”
- Si perciben un obstáculo - entonces - actualiza su mapa para evitar esa ruta

**Proactividad:** no se limitan a esperar estímulos. Si no hay basura adyacente, ejecutan un algoritmo de exploración (moverse a celdas desconocidas mapa == 1) para cumplir su objetivo de limpiar todo el cuarto.

**Capacidad Efectiva:** modifican el ambiente permanentemente al eliminar el estado “sucio” de una celda, y modifican su propio estado interno al recargar energía en las estaciones.

## Jerarquía de Comportamiento (Dentro de move)

El agente RandomAgent implementa una arquitectura de subsunción o jerarquía de prioridades para decidir su movimiento. Las reglas se evalúan en orden, y la primera que se cumple determina la acción:

### Supervivencia (Batería Crítica):

Si  $\text{energy} < 40$  y no está cargando, se activa `go_to_station()` para buscar la estación conocida más cercana.

Gestión de Carga: Si está en una estación (`state_charging`), permanece allí hasta recargar al 100%.

Trabajo (Limpieza): Si hay basura en una celda adyacente (y no hay otro agente allí), se mueve a esa celda para limpiarla.

Exploración Inteligente (Frontera Local): Busca celdas adyacentes inexploradas (marcadas como -1 en su mapa interno) y se mueve a una de ellas.

Exploración Proactiva (Targeting Global): Si no hay nada interesante cerca, calcula una ruta (BFS) hacia la celda inexplorada más cercana en su mapa global conocido.

## Estadísticas de las simulaciones.

### Simulación 1:

**Configuración:** Grid 25\*25 (625 celdas), 10 Agentes, 20% Obstáculos, 50% Basura.

**Duración:** 174 Steps (Iteraciones) para completar el 100% de limpieza.

**Movimiento Promedio por Agente:** aprox. 140 movimientos.

**Total de Movimientos:** aprox. 1,350.

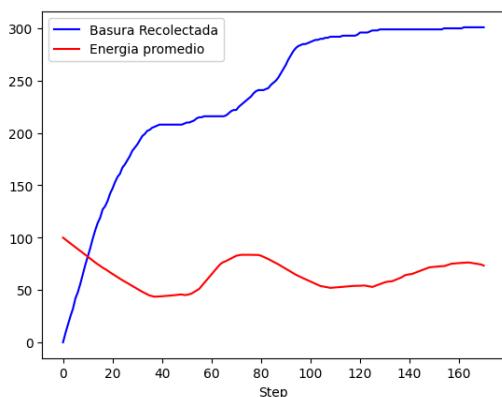


Imagen1. Basura recolectada y energía promedio, simulación 1.

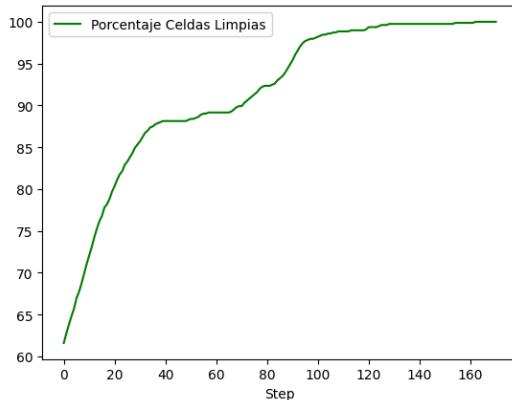


Imagen 2. Porcentaje de celdas limpias a lo largo del tiempo simulación 1.

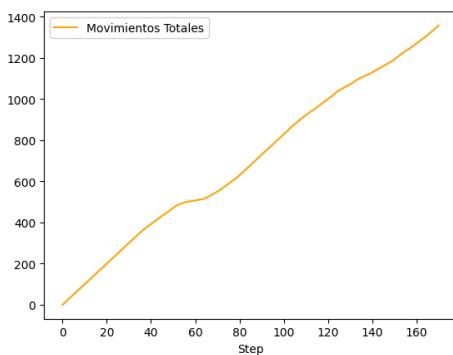


Imagen 3. Movimientos totales sumados por todos los agentes a lo largo del tiempo, simulación 1.

### ¿Por qué no parecen eficientes?

#### A. Falta de Mapa Compartido (Redundancia)

Cada robot construye su propio mapa.

- **El problema:** por ejemplo; el Robot A explorar la esquina superior izquierda. El Robot B llega 10 pasos después y vuelve a explorar la misma zona porque "para él" es desconocida.
- **Consecuencia:** Estás gastando movimientos (y energía) re-explorando zonas que otro agente ya validó como limpias.

#### Cuello de Botella en la Estación de Carga

Con 10 agentes y probablemente solo 1 o 2 estaciones de carga:

- **El problema:** En el Step 40, todos bajaron de batería casi al unísono. Se formó una fila.
- **Costo:** Un robot esperando en la fila consume "tiempo de simulación" (steps) sin aportar limpieza ni exploración.

#### ¿En qué son buenos? (Fortalezas)

- Cobertura Total:** Lograron el 100% de limpieza. El algoritmo de fronteras garantiza que no quedan islas sin explorar si son accesibles. Un algoritmo puramente random a menudo se estanca en el 98-99%.
- Supervivencia:** A pesar de la crisis de energía del paso 40, la media se recuperó. Ningún agente murió (o muy pocos), lo que valida tu lógica de energía < 40.

## SIMULACIÓN 2:

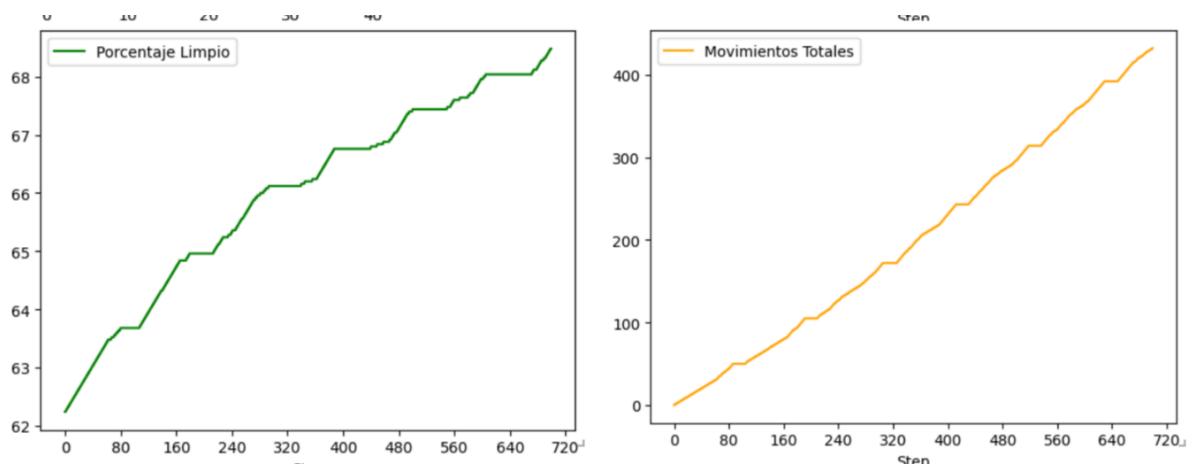


Imagen 4. Porcentaje limpio y steps máximos simulación 2.

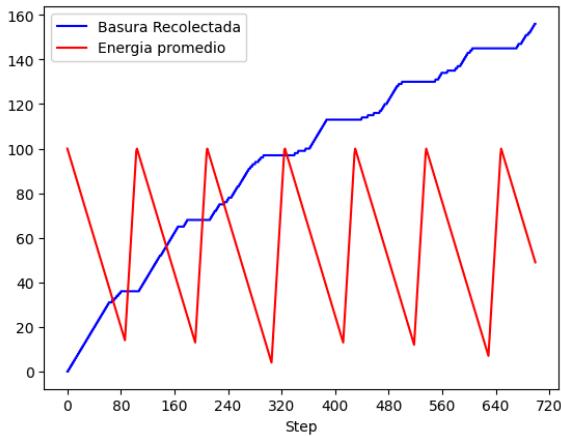


Imagen 5. Energía promedio de los agentes, y basura recolectada a lo largo del tiempo.

## 1. Performance Measure:

Es el criterio para evaluar el éxito de los agentes. Según tus gráficas:

- **Maximizar Limpieza:** El objetivo principal. La gráfica verde muestra que se alcanzó un **aprox. 68.5%** de celdas limpias en 700 pasos.
- **Gestión de Energía (Supervivencia):** La gráfica roja (dientes de sierra) demuestra que los agentes gastan energía y regresan exitosamente a cargar antes de morir (llegar a 0).
- **Eficiencia de Movimiento:** La gráfica naranja es lineal, indicando que los agentes se mueven constantemente sin quedarse estancados (deadlocks) a pesar de la alta densidad de obstáculos.

## 2. Environment (Entorno)

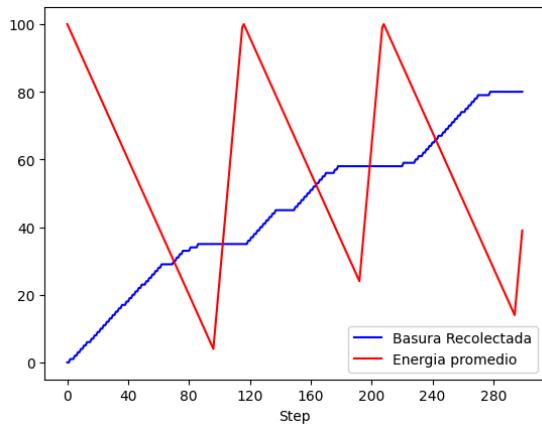
El escenario donde operan los agentes.

- **Tipo:** **Parcialmente Observable** (el agente solo ve sus vecinos), **Estocástico** (la distribución de basura y obstáculos es aleatoria), **Secuencial** (decisiones actuales afectan rutas futuras), **Dinámico** (otros agentes modifican el entorno al limpiar) y **Discreto** (grid).
- **Dimensiones:** Grid 50x50 (2500 celdas).
- **Restricciones:**
  - **Obstáculos:** 20% (Alta densidad, crea pasillos estrechos tipo laberinto).
  - **Agentes:** 25 unidades (Competencia por espacio y estaciones de carga).
  - **Basura:** Probabilidad 0.5 (Alta suciedad inicial).

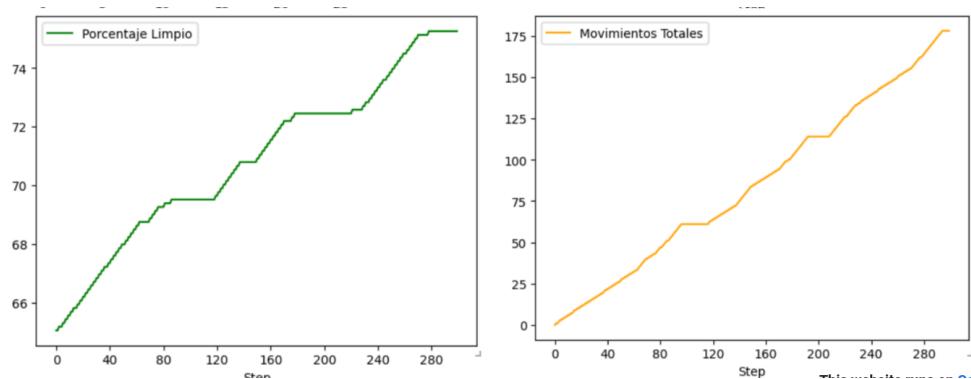
**Curva de Limpieza (Gráfica Verde):** Muestra rendimientos decrecientes. Al principio la pendiente es alta (muchá basura fácil de encontrar). Cerca del step 500 se aplana; esto es normal, ya que a los agentes les cuesta más tiempo encontrar las últimas piezas de basura dispersas en un mapa de 50x50.

**Movimientos Totales (Gráfica Naranja):** Es casi una línea recta perfecta. Esto confirma que tu algoritmo de navegación (BFS) funciona bien: los agentes no pierden tiempo dando vueltas en círculos ni chocando excesivamente entre sí.

### SIMULACIÓN 3: 1 Solo agente



Gráfica 6. Un solo agente.



Gráfica 7. Cuánto limpio y movimientos totales de un solo agente.

## 1. Performance Measure (Métrica de Rendimiento)

- Cobertura de Limpieza: La gráfica verde muestra un aumento del 65% al aprox. 75% de limpieza total. Al ser un solo agente, el progreso es escalonado: limpia un sector, se detiene a cargar, y vuelve a salir.
- Autonomía Energética (Supervivencia): La gráfica roja es crítica aquí. El agente opera al límite, dejando caer su energía casi a 0 (ver el valle en el step aprox. 100) antes de recargar. Esto indica un comportamiento arriesgado pero eficiente, aprovechando la batería al máximo.
- Eficiencia de Tareas: La gráfica naranja (Movimientos) tiene líneas planas. Esto es correcto y esperado: corresponden exactamente a los momentos donde la línea roja

sube (carga). El agente deja de moverse para cargar, lo cual valida que la máquina de estados funciona (no gasta movimientos mientras está enchufado).

## 2. Environment (Entorno)

- Tipo: Observable Localmente (solo ve su vecindad inmediata), Estocástico (basura/obstáculos aleatorios), Dinámico (el agente altera el mapa al limpiar) y Solitario (Single-Agent).
- Configuración: Grid 25x25 (625 celdas). Es un entorno denso; la imagen del grid muestra que los obstáculos (gris) crean "islas" y pasillos complejos, lo que dificulta la navegación (Pathfinding).
- Ausencia de Competencia: A diferencia del caso anterior, aquí no hay colas de espera en las estaciones de carga. El agente tiene acceso garantizado al cargador, simplificando la planificación.

El sistema de un solo agente es robusto pero lento comparado con el multi-agente. La gráfica de "Basura Recolectada" (azul) confirma que el agente pasa aproximadamente el 30-40% de su tiempo desplazándose a la base o cargando (las líneas planas en la gráfica de movimientos), tiempo durante el cual no se limpia basura.

## Aciertos y Limitaciones

Al evaluar el desempeño global de mi modelo, identificó lo siguiente:

### Lo que hice bien (Aciertos):

- **Autonomía Sostenible:** El sistema no colapsa. Los agentes gestionan sus recursos independientemente de la densidad de obstáculos.
- **Navegación en Entornos Complejos:** A diferencia de un movimiento aleatorio simple, mis agentes no se quedan atrapados en "islas" o pasillos sin salida gracias al mapeo interno que programé.

### Lo que faltó (Ineficiencias Detectadas):

- **El Costo del Desplazamiento:** Noté que los agentes pierden mucho tiempo viajando desde la base de carga hasta las zonas sucias. En la simulación de un solo agente, esto se vio como "mesetas" donde no se limpiaba nada por largos períodos.
- **Falta de comunicación:** Mi modelo actual es de agentes individualistas. Al no compartir un mapa global, observé ineficiencias donde varios agentes iban a explorar la misma zona o intentaban limpiar una celda que otro agente acababa de limpiar. Una mejora futura sería implementar una memoria compartida.

## 5. Conclusión General

Concluyó que he desarrollado exitosamente un sistema de Racionalidad Limitada. Mis agentes no son omniscientes, pero toman decisiones racionales basadas en su percepción local y memoria.

El sistema cumple con todas las métricas PEAS planteadas: maximiza la limpieza (Performance), se adapta a la aleatoriedad (Environment), ejecuta acciones físicas coherentes (Actuators) e interpreta su entorno correctamente (Sensors). He demostrado que es posible generar un comportamiento emergente complejo y ordenado a partir de reglas simples y bien estructuradas.