M1 - Actividad

Sebastian Alvarez Fuentes A01800280

Miguel Angel Argumedo Ortiz A01751322

Isaac Calderon Laflor A01751722

Ingeniería en Tecnologías Computacionales, Tecnológico de Monterrey

Modelación de sistemas multiagentes con gráficas computacionales

Jorge Adolfo Ramirez Uresti

14 de Noviembre 2025

Introducción

Objetivo general de la actividad:

Analizar el comportamiento de un sistema multiagente de limpieza reactivo en una cuadrícula MxN, midiendo cómo influyen el número de agentes y los parámetros del entorno en el tiempo de limpieza, los movimientos realizados y el porcentaje final de celdas limpias.

Objetivos específicos:

1. Implementar en Python, usando la librería Mesa, un modelo multiagente de agentes aspiradores que limpian celdas sucias en un grid MxN.
2. Diseñar y ejecutar experimentos controlados variando el número de agentes, el porcentaje inicial de suciedad y el tiempo máximo de simulación.
3. Registrar y analizar las estadísticas de desempeño (tiempo, movimientos y porcentaje de limpieza) para identificar patrones de rendimiento y posibles comportamientos emergentes del sistema.

Descripción del problema

Notas adicionales del problema:

Se utilizó la librería Mesa 3.3.1 para modelar el entorno y los agentes de limpieza. El movimiento se define sobre una vecindad de Moore de radio 1, sin considerar bordes toroidales (la cuadrícula tiene límites duros). La simulación está acotada por un número máximo de pasos definidos por el usuario, incluso si aún quedan celdas sucias.

Diseño del modelo

Descripción del comportamiento del agente en cada paso:

En cada paso el agente observa la celda en la que se encuentra. Si la celda está marcada como sucia, la limpia (la cambia a estado limpio) y termina su acción en ese tick. Si la celda ya está limpia, obtiene la lista de celdas vecinas usando vecindad de Moore de radio 1, excluyendo su propia celda y descartando las posiciones fuera de los límites del grid. De entre las celdas vecinas válidas selecciona una al azar; si existe al menos una, se mueve a esa posición e incrementa su contador de movimientos. Si no hay vecinas válidas, permanece en su celda actual.

Descripción del proceso de inicialización del modelo:

Al crear el modelo se construye una cuadrícula MultiGrid de tamaño MxN sin toroidal. A partir del porcentaje de suciedad especificado se calcula cuántas celdas deben inicializarse sucias y se seleccionan posiciones aleatorias distintas para marcarlas como tales. Después se instancia el número de agentes indicado por el usuario, asignándoles identificadores consecutivos y colocándolos a todos en la posición inicial (1,1) de la cuadrícula. Finalmente se inicializan los contadores globales de la simulación (número de pasos y movimientos totales) y se deja el modelo listo para comenzar a ejecutar los pasos de la dinámica.

Metodología experimental

Tamaño de la cuadrícula:

- M (filas): 10
- N (columnas): 10

Porcentaje de celdas sucias iniciales: 40 %

Tiempo máximo de ejecución (en pasos): 100

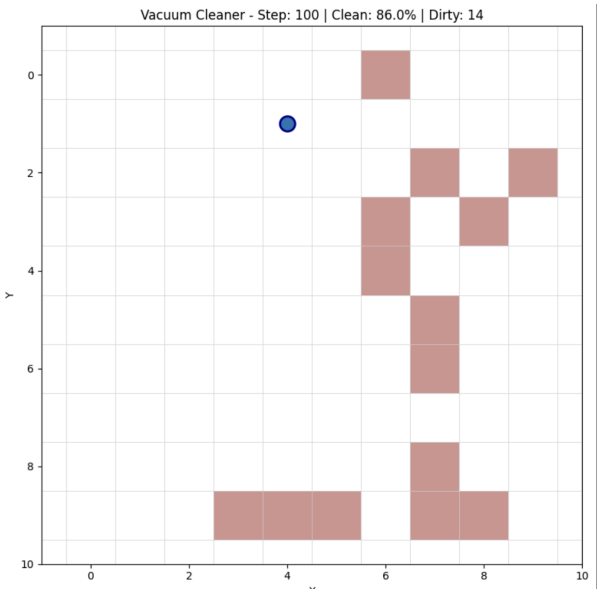
Valores de número de agentes probados:

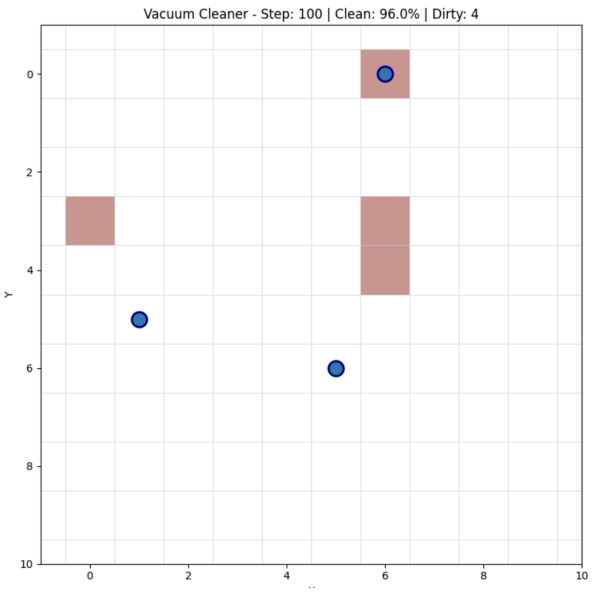
- **Caso 1:** 1 agentes
- **Caso 2:** 5 agentes
- **Caso 3:** 10 agentes
- **Caso 4:** 25 agentes

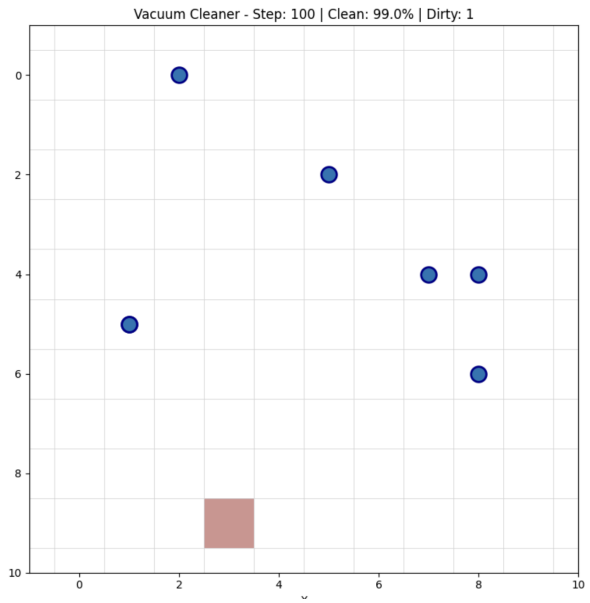
Resultado

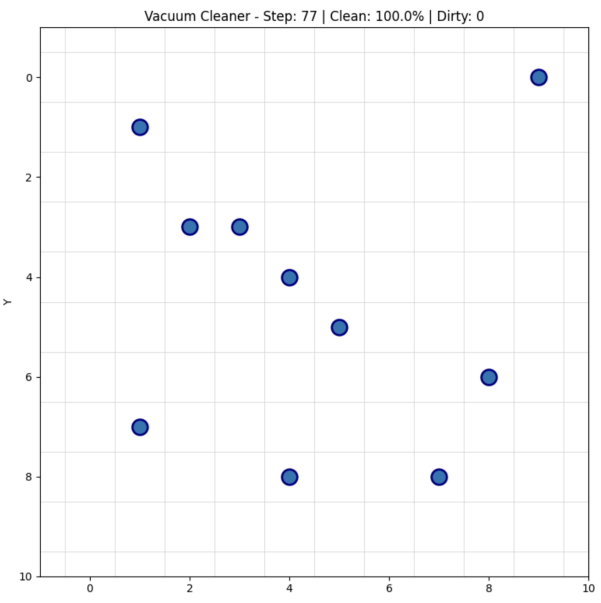
Tabla 1. Resultados por número de agentes

Agentes	Tiempo total (pasos)	% limpio final
1	100	86
3	100	96
7	100	99
10	77	100









Análisis de resultados

El experimento se ejecutó en una cuadrícula 10 x 10 con 40% de suciedad inicial y un límite de 100 pasos. Los resultados muestran una clara dependencia del rendimiento respecto al número de agentes.

El porcentaje de limpieza logrado es la métrica que mejora consistentemente: sube de 86% con 1 agente a 100% con 10 agentes. El aumento en la cantidad de agentes incrementa la cobertura y la capacidad del sistema para encontrar la suciedad antes de que el tiempo se acabe.

Respecto al tiempo total de limpieza, se observó que agregar más agentes no siempre reduce el tiempo. Con 1, 3 y 7 agentes, la simulación falló en limpiar todo antes de llegar al límite de 100 pasos. La reducción del tiempo total de finalización solo se logró al alcanzar los 10 agentes (77 pasos), lo que demuestra la necesidad de una "masa crítica" para que la colaboración supere la ineficiencia.

Aunque el tiempo de finalización baja al tener 10 agentes, el número total de movimientos del sistema probablemente sube. Esto nos lleva a cuestionar la eficiencia: ¿Existe un punto donde agregar más agentes deja de ser rentable? Sí, es muy probable. En una cuadrícula pequeña (10 x 10), seguir añadiendo agentes causaría congestión y haría que el sistema dedicara la mayor parte de su tiempo a realizar movimientos inútiles en celdas ya limpias, aumentando el costo de operación sin una mejora proporcional en el tiempo.

El principal problema son los comportamientos emergentes ineficientes. El agente es reactivo puro: no tiene memoria ni inteligencia. Esta estupidez sistémica lleva a una redundancia de exploración masiva, donde los agentes visitan celdas limpias repetidamente, y compiten por el acceso a la suciedad restante.

Las limitaciones clave del modelo son precisamente la falta de inteligencia de los agentes, la mala inicialización (todos parten de (1,1) causando congestión inicial) y el límite de 100 pasos que actúa como un tope artificial.

Conclusiones

El análisis del sistema multiagente reactivo demostró que la **cantidad de agentes es el factor determinante** para alcanzar una **cobertura de limpieza completa** dentro de un tiempo límite. Los resultados confirmaron que la eficiencia no es lineal: se requiere una **"masa crítica" de agentes (10 agentes)** para que la ventaja de la paralelización supere las ineficiencias y logre una reducción efectiva en el tiempo total de finalización de la tarea (77 pasos).

Sin embargo, el modelo evidenció una **alta ineficiencia** intrínseca. Los agentes reactivos, al carecer de **memoria** y depender de un movimiento **puramente aleatorio**, incurren en **redundancia de exploración masiva**, visitando celdas ya limpias repetidamente. Este comportamiento, sumado a la **congestión inicial** por la colocación de todos los agentes en (1,1), sugiere que, a pesar de la rápida limpieza, el **costo total de movimientos del sistema probablemente se dispara**.

En resumen, aunque la solución multiagente mejora drásticamente el **porcentaje de limpieza** y la **velocidad de finalización** (a partir del umbral óptimo), el modelo está limitado por la falta de un comportamiento inteligente. Las futuras iteraciones deberían enfocarse en dotar a los agentes de **algoritmos de búsqueda heurística o memoria** para minimizar la exploración redundante y optimizar la relación entre el número de agentes, el tiempo de limpieza y el costo de movimientos.

Código fuente y repositorio

Enlace al repositorio de GitHub: https://github.com/mikeargu/M1-Actividad