Tecnológico de Monterrey

TC2008B TEC-M1

TC2008B Modelación de sistemas multiagentes con gráficas computacionales

Estadísticas de un robot de limpieza reactivo.

Reporte de Actividad

Tecnológico de Monterrey
Alumna: Mariana Isabel Aguilar Espíndola

Matrícula: A01772501

Contenido

1. Resumen

El presente documento aborda el estudio de un sistema de limpieza reactivo mediante simulación computacional. El objetivo principal consiste en analizar el comportamiento y rendimiento de uno o varios agentes de limpieza, evaluando cómo la cantidad de agentes impacta el tiempo total de operación y el número de movimientos realizados durante el proceso de limpieza.

Para ello, se consideran dos modos de exploración distintos: **exploración aleatoria** y **exploración planeada**. En el modo de exploración aleatoria, los agentes se desplazan en direcciones al azar, eligiendo posibles nodos dentro de la matriz con el fin de localizar y eliminar la basura. En contraste, la exploración planeada se basa en trayectorias previamente definidas que buscan optimizar la cobertura del entorno, reduciendo el número de movimientos redundantes.

A lo largo del informe se presentan diversos experimentos que permiten comparar ambos métodos de exploración, observando las diferencias en el tiempo total de simulación, el porcentaje de celdas limpias al finalizar la ejecución y la cantidad total de movimientos realizados. Los resultados obtenidos proporcionan una visión cuantitativa del desempeño de los agentes de limpieza y de la influencia del modo de exploración en la eficiencia global del sistema.

2. Introducción

Un *robot de limpieza reactivo* es un agente autónomo que toma decisiones de forma inmediata basándose únicamente en la información sensorial que percibe en tiempo real, sin disponer de memoria del entorno ni de un modelo previo del mismo. Este tipo de robot no sigue un plan predefinido, sino que adapta su comportamiento dinámicamente en función del estado actual del entorno. Así, cuando se encuentra frente a un obstáculo o una celda sucia, actúa de acuerdo con reglas simples, como aspirar la suciedad presente.

En el contexto de este estudio, los agentes de limpieza implementan un comportamiento completamente reactivo: carecen de memoria y no se comunican entre sí. Su única regla de actuación consiste en aspirar cuando se encuentran sobre una celda sucia y llevarla al bote de basura, en caso contrario deben desplazarse de manera aleatoria hacia una de las celdas vecinas disponibles. Este tipo de sistema, aunque simple, permite analizar la eficiencia en la que varios agentes reactivos trabajan.

El presente trabajo tiene como objetivo evaluar la eficiencia del proceso de limpieza bajo diferentes parámetros, tales como el número de agentes en operación, el porcentaje inicial de suciedad y el tiempo máximo de ejecución, etc. Para ello, se realizan múltiples simulaciones, en las cuales se recopilan métricas clave.

Finalmente, se analiza cómo la variación en el número de agentes influye en el desempeño global del sistema, considerando tanto el tiempo invertido como el esfuerzo computacional asociado al desplazamiento colectivo. Este estudio permite comprender de manera más profunda el comportamiento de los sistemas reactivos distribuidos y su potencial para resolver tareas de limpieza de manera eficiente sin requerir planificación centralizada.

2.1. Objetivos

- Implementar y comparar dos estrategias de exploración: aleatoria y planeada
- Analizar el impacto del número de agentes en el tiempo de limpieza
- Evaluar la eficiencia en términos de movimientos realizados
- Determinar el porcentaje de limpieza alcanzado en tiempo limitado

3. Descripción de archivos principales

La implementación se compone de varios scripts principales:

- `Lib_TC2008.py`: contiene la inicialización del entorno gráfico (Pygame + OpenGL), la carga de parámetros desde `Settings.yaml`, la creación de agentes (`Lifter`) y objetos de suciedad (`Basura`), y el bucle principal de simulación (`Simulacion`).
- `Lifter.py`: define la clase `Lifter`, que modela el agente móvil, con su lógica de movimiento planeado o aleatorio, control de estados (`searching`, `lifting`, `delivering`, `dropping`, ...) y la actualización física/animación.
- `Main.py`: define los parsers que se pueden usar en la simulación.
- `Basura.py`: clase que representa los objetos a recoger, considerados como basura.
- `Datos/`: carpeta en la que se guardan datos relevantes de la simulación en formato CSV.

3.1. Parámetros de Simulación

Al momento de realizar la simulación se pueden controlar los siguientes parámetros según sea necesario por medio de un parser:

- **M**: Tamaño de la matriz para la ruta aleatoria ($M \times M$)
- **N**: Número de agentes (robots de limpieza)
- **Basuras**: Número de basuras ubicadas aleatoriamente
- **T_max**: Tiempo máximo de ejecución
- **Método**: Exploración aleatoria o planeada

3.2. Parser

Se implementó un parser en la línea de comandos que permite configurar los parámetros anteriormente mencionados de la simulación

- `--lifters`: número de agentes (obligatorio).
- `--Basuras`: número de objetos de basura (obligatorio).
- `--Delta`: velocidad de simulación (opcional, valor predeterminado: 0.05).
- `--theta`, `--radious`: parámetros gráficos (opcional).
- `--method`: método de navegación: `planned` o `random` (predeterminado: `planned`).
- `--Tmax`: duración máxima de la simulación en segundos (predeterminado: 60).
- `--M`: tamaño de la matriz $M \times M$ para el método aleatorio (predeterminado: 5).

4. Metodología

4.1. Estrategias de navegación

En cada iteración del bucle principal, el agente se desplaza por la matriz según la estrategia de navegación seleccionada: **Exploración planeada**: En este modo, siempre se utiliza una matriz de 5×5, ya que las rutas planeadas están diseñadas

específicamente para cubrir este tamaño. El valor del parámetro M especificado en el parser se ignora, pues únicamente es válido en la exploración aleatoria. Como se muestra en la Figura 1, dependiendo del número de *lifters* especificado,

se asigna una ruta específica a cada agente. Estas rutas están diseñadas para cubrir toda la matriz sin superposiciones innecesarias. Si un *lifter* encuentra basura en su trayectoria, la recoge y la lleva al nodo 0 (donde se ubica el bote de basura), la deposita y retoma su ruta desde el nodo en el que se interrumpió. Una vez que un agente completa su recorrido, regresa al nodo 0 y permanece allí hasta que finalice la simulación o todos los agentes hayan terminado. Dado que las rutas están diseñadas para cubrir completamente la matriz de 5×5, con una sola ejecución se garantiza que no quede basura en la sección asignada.

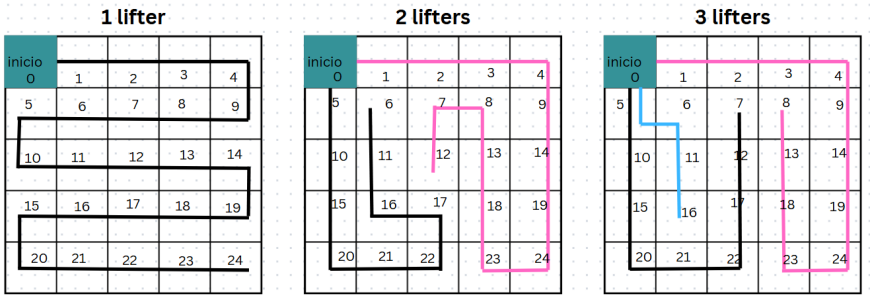

Figura 1: Diseño de rutas por lifter

Se consideran tres configuraciones según el número de agentes:

- **1 lifter**: recorre toda la matriz siguiendo un patrón en zigzag.
- **2 lifters**: cada uno recorre aproximadamente la mitad de la matriz mediante rutas separadas.
- **3 lifters**: cada agente sigue una ruta más corta pero eficiente, evitando colisiones entre ellos.

Exploración aleatoria: En este modo, el tamaño de la matriz ($M \times M$) se define mediante el parámetro `--M`. A partir de este valor, se genera automáticamente una matriz de adyacencia que indica qué nodos están conectados. Los agentes no siguen un orden predefinido; en cada paso, examinan los nodos vecinos accesibles y eligen uno al azar para desplazarse. Si durante su trayecto encuentran basura, la recogen y la llevan al bote de basura (nodo 0) antes de continuar su exploración. Este enfoque simula un comportamiento puramente reactivo, sin planificación ni coordinación explícita entre agentes.

4.2. Datos Recopilados en Csv

En cada ejecución se genera un archivo csv que contiene información relevante de cada agente. Se crea un archivo por agente, en donde el nombre del archivo menciona el nombre del agente, la fecha y hora de la simulación y el método que se utilizó. Los datos que se pueden visualizar son:

- **TimeStamp**: Momento en el que se realizó una acción en formato HH:MM:SS.microsegundos.
- **AgentID**: Identificador del agente en la simulación, empieza en 0 y dependiendo el número de agentes va subiendo
- **State**: estado del agente,permite observar si está buscando o recogiendo alguna basura
- **PosX, PosY, PosZ**: posición en el plano 3D del agente
- **CurrentNode**: nodo en el que el agente se encuentra
- **NextNode**: siguiente nodo al cual el agente deberá ir
- **Ruta**: muestra la ruta planeada que tiene el agente o si es random

```
timestamp,agent_id,method,state,pos_x,pos_y,pos_z,current_node,next_node,route
12:47:28.842,2,planned,searching,0.0,0.0,0.7,0,5,"[0, 5, 6, 11, 16, 0]"
12:47:29.680,2,planned,delivering,0.0,0.0,15.4,0,5,"[0, 5, 6, 11, 16, 0]"
12:47:29.980,2,planned,dropping,0.0,0.0,9.8,0,5,"[0, 5, 6, 11, 16, 0]"
12:47:30.126,2,planned,searching,0.0,0.0,9.8,0,0,"[0, 5, 6, 11, 16, 0]"
12:47:31.392,2,planned,delivering,0.0,0.0,15.4,0,5,"[0, 5, 6, 11, 16, 0]"
12:47:31.703,2,planned,dropping,0.0,0.0,9.8,0,5,"[0, 5, 6, 11, 16, 0]"
12:47:31.828,2,planned,searching,0.0,0.0,9.8,0,0,"[0, 5, 6, 11, 16, 0]"
12:47:33.498,2,planned,delivering,16.62,0.0,16.74,0,6,"[0, 5, 6, 11, 16, 0]"
12:47:33.998,2,planned,dropping,9.72,0.0,9.79,0,6,"[0, 5, 6, 11, 16, 0]"
12:47:34.132,2,planned,searching,9.72,0.0,9.79,0,0,"[0, 5, 6, 11, 16, 0]"
12:47:36.758,2,planned,delivering,18.07,0.0,36.06,0,11,"[0, 5, 6, 11, 16, 0]"
12:47:38.202,2,planned,dropping,4.9,0.0,9.77,0,11,"[0, 5, 6, 11, 16, 0]"
12:47:38.336,2,planned,searching,4.9,0.0,9.77,0,0,"[0, 5, 6, 11, 16, 0]"
12:47:41.687,2,planned,delivering,18.52,0.0,55.51,0,16,"[0, 5, 6, 11, 16, 0]"
```

Figura 2: Ejemplo de datos de CSV por agente.

4.3. Resumen de la Simulación

El parser tiene la opción de permitir visualizar un resumen de la simulación en donde se muestran de manera resumida datos de interés.

- **Tipo de experimento**: planeado o aleatorio
- **Tmax**: tiempo máximo que correrá la simulación
- **M**:tamaño de la matriz
- **N**: número de agentes
- **Celdas inicialmente sucias**: porcentaje de celdas
- **Celdas limpias al final**: porcentaje de celdas limpias al finalizar
- **Tiempo real de simulación**: cuanto tiempo de la simulación realmente tuvo que utilizar para limpiar
- **Movimientos totales**: cuantas veces cambio de nodo

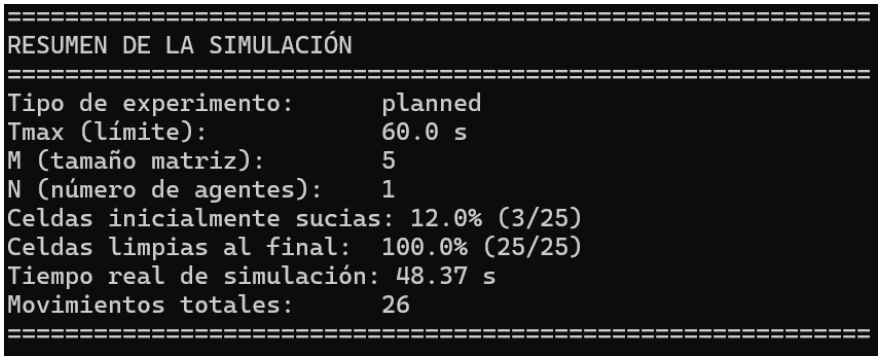

Figura 3: Ejemplo de tabla generada al finalizar la simulación

5. Resultados

A continuación se presenta un resumen de 10 experimentos realizados en donde se cambia el número de agentes, basuras y el método utilizado

TABLA. 1
Resumen de experimentos realizados.

Exp	Tipo	M	N	% Inicial sucio	% Final limpio	T_{max} (s)	Tiempo (s)	Movimientos
1	planned	5	1	12.0	100.0	60.0	55.89	26
2	planned	5	1	20.0	100.0	60.0	52.60	26
3	planned	5	2	32.0	96.0	60.0	60.85	35
4	planned	5	3	16.0	100.0	60.0	55.89	46
5	planned	5	3	16.0	100.0	40.0	33.99	43
6	random	5	3	16.0	96.0	40.0	40.73	98
7	random	5	1	24.0	88.0	40.0	40.67	26
8	random	5	6	32.0	100.0	60.0	60.44	277
9	random	8	6	12.5	95.3	60.0	66.21	284
10	random	8	6	1.6	100.0	60.0	25.32	289

6. Análisis

6.1. Análisis del impacto de la cantidad de agentes en el desempeño

En los experimentos realizados se observa una clara relación entre la cantidad de agentes y el desempeño general de la simulación, tanto en términos de tiempo de ejecución como en la cantidad de movimientos requeridos para alcanzar los objetivos de limpieza. A continuación, se analizan los resultados obtenidos bajo los dos modos de exploración: *planeado* y *aleatorio*.

6.2. Método planeado

En el método *planeado*, los agentes siguen trayectorias planificadas y coordinadas. Los resultados muestran que, al aumentar el número de agentes, el tiempo real de simulación tiende a mantenerse cercano al límite máximo establecido, aunque se observa una ligera mejora en los casos con tres agentes. Por ejemplo, con un agente y un 20% de suciedad inicial, el sistema logra una limpieza total en aproximadamente 52.6 segundos y 26 movimientos. En cambio, con tres agentes, el tiempo se reduce a 33.99 segundos, logrando también una limpieza del 100%.

Este comportamiento indica que la colaboración entre múltiples agentes planificados puede distribuir las tareas de manera eficiente, reduciendo tanto el tiempo de ejecución como la redundancia en los movimientos. Sin embargo, el incremento del número de agentes no necesariamente implica una reducción lineal del tiempo, debido a la posible superposición de trayectorias y la coordinación necesaria entre ellos.

6.3. Método aleatorio

En el método *aleatorio*, los agentes se desplazan de manera no planificada, lo que genera un comportamiento menos eficiente. Los resultados reflejan un mayor número de movimientos y, en algunos casos, un menor porcentaje de limpieza alcanzado dentro del tiempo máximo.

Por ejemplo, con un agente, el sistema alcanza sólo un 88% de limpieza en 40.67 segundos y realiza 26 movimientos, mientras que con tres agentes, la limpieza mejora a un 96%, pero el número de movimientos aumenta significativamente hasta 98. Con seis agentes en un entorno de 5x5, se obtiene un 100% de limpieza, aunque con un costo elevado de 277 movimientos.

Esto sugiere que, en el modo aleatorio, incrementar el número de agentes mejora la cobertura espacial, pero también incrementa el número total de movimientos debido a la falta de coordinación entre los mismos. A medida que el entorno crece (por ejemplo, una matriz de 8x8), los resultados muestran que la eficiencia depende tanto del número de agentes como de la cantidad de suciedad inicial.

7. Conclusión

Comparando ambos modos, se puede afirmar que el método *planeado* ofrece un mejor equilibrio entre tiempo de simulación y número de movimientos, mostrando una limpieza más eficiente con menor esfuerzo total. En contraste, el método *aleatorio* logra resultados aceptables, pero a costa de una mayor cantidad de movimientos y tiempos más variables. Por lo tanto, la planificación y coordinación entre agentes resultan determinantes para optimizar el rendimiento del sistema de limpieza multiagente.

Código

Link al zip del código