Tecnológico de Monterrey

TC2008B Modelación de sistemas multiagentes con gráficas computacionales

Estadísticas de un robot de limpieza reactivo

Reporte de Actividad

Alumnos:
Pedro Enrique Mendoza García
Brisa Sofía Leon Pérez
Valentina Tejeda Fuentes
Mariana Isabel Aguilar Espíndola
Leobardo de Jesús Carbajal
Anairam San Nicolás Rodriguez

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`: procesa la configuración de la simulación. Permite dos fuentes:
  - argumentos desde línea de comandos (parser),
  - archivo JSON que contiene los parámetros completos.

  Si ambos se proporcionan, los argumentos del parser sobrescriben a los del JSON.
- `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

La simulación puede configurarse mediante:

- **Argumentos en línea de comandos**
- **Archivo JSON**

Si ambos se usan simultáneamente, los argumentos del parser tienen prioridad.
Los parámetros disponibles son:

- `--config`: archivo JSON con la configuración.
- `--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).

3.2. Configuración mediante archivo JSON

Además del parser, la simulación admite un archivo JSON desde el cual se cargan todos los parámetros. Ejemplo de configuración:

```
{
  "command": "Simulacion",
  "lifters": 3,
  "basuras": 6,
  "method": "random",
  "Tmax": 60,
  "M": 5,
  "Delta": 0.05,
  "theta": 0,
  "radious": 30,
  "resumen": "s"
}
```

Los parámetros definidos en la línea de comandos sobrescriben a los del archivo JSON.

4. Ejemplos de uso

A continuación se muestran algunos ejemplos básicos para ejecutar la simulación empleando argumentos de línea de comandos o un archivo de configuración JSON.

Simulación con método aleatorio

```
python Main.py Simulacion --lifters 3 --Basuras 6 --method random --Tmax 60 --M 5
```

Simulación con método planeado

```
python Main.py Simulacion --lifters 2 --Basuras 4 --method planned --Tmax 60
```

Usando archivo JSON

```
python Main.py Simulacion --config config_example.json
```

5. Metodología

5.1. Estrategias de navegación

Cada agente se desplaza según el método especificado:

Exploración planeada

Las rutas planeadas fueron diseñadas para permitir:

- Cobertura completa del mapa independientemente del número de agentes.
- Evitar colisiones entre agentes usando zonas de seguridad.
- Reanudar la ruta exactamente donde se interrumpió al encontrar una basura.
- los agentes inician en nodos aleatorios automáticamente,
- el movimiento incorpora evasión de colisiones fuera del área de incineración.

Exploración aleatoria

Los agentes seleccionan un nodo vecino aleatorio en cada movimiento, integrando un sistema de evasión:

- si detectan otro agente cerca y están fuera del área de tirar basura, cambian su dirección,
- en la zona central (incinerador) se permite el amontonamiento,
- cada vez que un agente debe desviarse se incrementa su contador de interferencias.

El resumen final indica cuántos agentes entorpecen, separados por método (`planned` o `random`).

5.2. Determinación de agentes que entorpecen la tarea

Cada `Lifter` mantiene un contador de interferencias, el cual aumenta cuando el agente debe alterar su trayectoria por la presencia de otro agente.

Al finalizar la simulación se clasifican como agentes que **entorpecen la tarea** aquellos que cumplen:

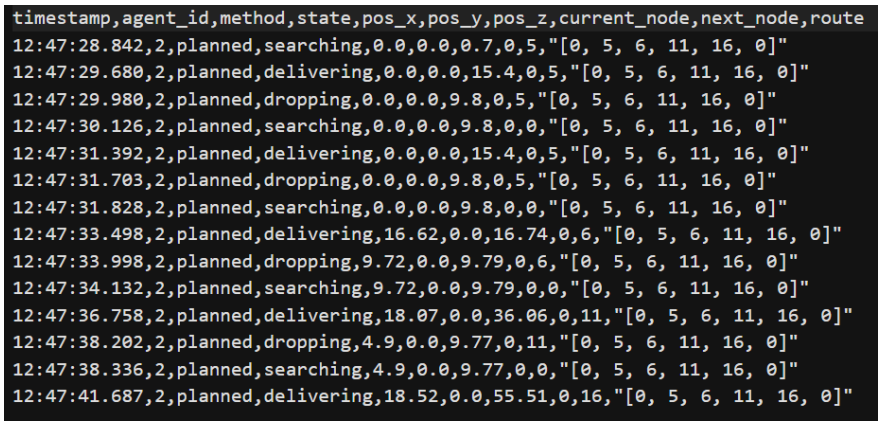
- presentan un número de interferencias superior al 15% del promedio global de movimientos, o
- acumulan cinco o más interferencias, aun si su número de movimientos fue bajo.

El resumen final indica cuántos agentes entorpecen, separados por método (`planned` o `random`).

5.3. 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 1: Ejemplo de datos de CSV por agente.

5.4. 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
- **U**mbral de interferencias calculado
- **N**úmero de agentes que entorpecen (total y por método)

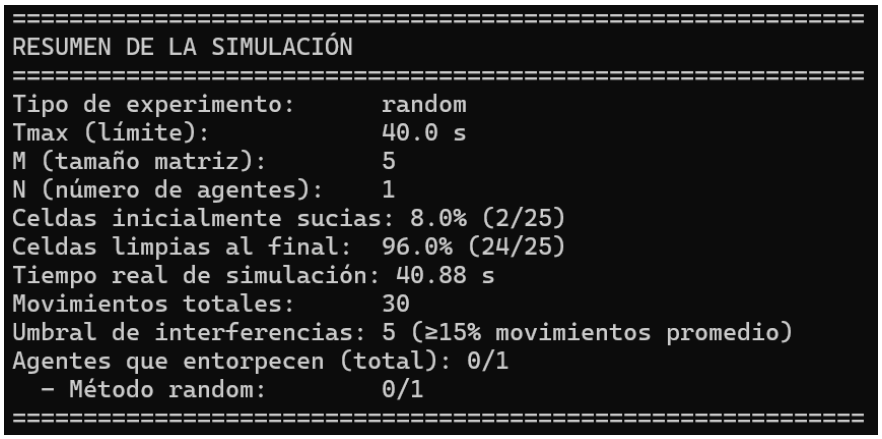


Figura 2: Ejemplo de tabla generada al finalizar la simulación

6. 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)	Movimientos	Umbral de interferencias	Agentes que entorpecen (total)
1	planned	5	1	8.0	100.0	40.0	27	5	0/1
2	planned	5	2	12.0	100.0	60.0	48	5	0/2
3	planned	5	3	20.0	96.0	60.0	42	5	2/3
4	planned	7	5	16.0	100.0	60.0	83	5	4/5
5	planned	8	6	17.0	98.0	40.0	120	5	5/6
6	random	5	1	8.0	96.0	40.0	33	5	0/1
7	random	5	2	12.0	100.0	60.0	86	6	1/2
8	random	5	3	20.0	100.0	60.0	127	6	3/3
9	random	7	5	16.0	100.0	60.0	92	5	5/5
10	random	8	6	17.0	98.0	40.0	91	5	5/6

7. Análisis

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

En los experimentos realizados, se observa cómo la cantidad de agentes y el método de exploración influyen tanto en la eficiencia de limpieza como en el número de movimientos realizados. La tabla de resultados muestra claramente que los métodos planeado y aleatorio tienen comportamientos distintos. A continuación, se analizan los resultados obtenidos bajo los dos modos de exploración: *planeado* y *aleatorio*.

7.2. Método planeado

Para los experimentos con el método planeado:

Con 1 agente y una suciedad inicial baja (8 - 12 porciento), el sistema logra limpiar el 100 porciento de las celdas en tiempos de 40–60 segundos con un número de movimientos relativamente bajo (27–48 movimientos).

A medida que se incrementa el número de agentes (3–6), los movimientos totales aumentan (42–120), pero se mantiene la limpieza completa en la mayoría de los casos. El número de agentes que entorpecen es bajo, en general 0–5, lo que refleja una buena coordinación entre los agentes planificados.

Esto indica que el método planeado permite distribuir eficientemente las tareas y mantener la limpieza casi completa, aunque los movimientos totales crecen con más agentes debido a la cobertura adicional requerida.

7.3. Método aleatorio

Para los experimentos con el método aleatorio:

Con pocos agentes (1–2), se observa un rendimiento inferior: la limpieza final varía entre 96 y 100 porciento, y los movimientos pueden ser relativamente altos incluso con pocos agentes (33–86 movimientos). Al aumentar el número de agentes (3–6), la limpieza final alcanza 100 porceinto, pero los movimientos totales crecen considerablemente (127–277 movimientos). El número de agentes que entorpecen es mayor que en el método planeado, llegando hasta 5–6 agentes en las configuraciones más grandes.

Esto evidencia que, aunque aumentar agentes mejora la cobertura espacial, la falta de coordinación en el método aleatorio genera un alto número de interferencias y movimientos redundantes.

8. Conclusión

Observando en general la eficiencia de limpieza, el método planeado mantiene la limpieza al 100 porciento con menos interferencias y movimientos totales más controlados. En el aspecto de movimientos y tiempo, el método aleatorio requiere muchos más movimientos para alcanzar resultados similares, y el tiempo real de simulación no siempre se beneficia de más agentes debido a las interferencias. Finalmente analizando a los agentes que entorpecen, el método planeado reduce significativamente la interferencia; en el método aleatorio, los agentes tienden a interferir más a medida que se incrementa el número de participantes.

En conclusión, los resultados confirman que la planificación es clave para optimizar la eficiencia del sistema multiagente. La tabla muestra claramente que el método planeado logra un mejor balance entre tiempo, número de movimientos y control de interferencias, mientras que el método aleatorio puede ser efectivo, pero con un mayor costo en eficiencia y coordinación.

Código

Link al repositorio de Github del código