# Modelación de Sistemas Multiagentes y Gráficas Computacionales

## Evidencia 2. Documento

Agosto de 2023

Integrantes

Diego Perdomo Salcedo

A01709150

Julio Cesár Perez Rodriguez

A01

Leslie del Carmen Sánchez Reyes

A01708987

# Idea inicial para el proyecto.

Al presentarnos con este reto discutimos en equipo como llegaríamos a resolver el problema. Tuvimos muchas ideas diferentes en la etapa de planeación. Consideramos que la mejor manera para que los robots limpien la oficina en menor tiempo era dividir la oficina en 5 zonas iguales, una para cada robot. De esta forma los robots rastrearían su área correspondiente y recogerían toda la basura de su zona. Si un robot termina de limpiar su zona se pasaría a la siguiente a ayudar a los demás robots. Al quedar conformes con esta idea intentamos programarla, pero se nos presentaban muchos errores en el camino. No sabíamos cómo dividir el modelo de la oficina en diferentes zonas correspondientes del tamaño de la misma. Lo cual después de algunos días de varios intentos decidimos probar con otra de nuestras ideas ya que esta no nos estaba dando el resultado esperado.

La siguiente idea que tuvimos fue que los robots empezarán en la parte de arriba del mapa separados por una cierta distancia, y luego bajarán hasta el fondo, se movieran un poco a la derecha y luego volvieran a subir. De esta forma todos mapearían el modelo de la oficina de una forma ordenada y uniforme. Pero similar al pasado también nos encontramos con problemas. No lograbamos dividir el espacio entre los robots correctamente lo cual causaba muchos problemas al mapear el mapa de la manera más eficiente.

Como resultado decidimos llegar a una nueva idea que nos permitió recorrer el mapa de una forma relativamente más eficiente y fácil de ejecutar.

# La solución implementada (Conceptualmente).

Esta solución es relativamente simple. Se intentará describir en tres pasos:

- Paso 1: Al inicio cada robot tiene su propio destino. 4 de los robots se van a cada una de las esquinas de la oficina, y el otro va al centro. Mientras llegan a su posición van mapeando el área que tienen.

- Paso 2: Van a empezar a mapear el área buscando en la matriz las posiciones que no han encontrado. Estas posiciones están definidas con un valor 0 así que seguirá mapeando hasta que no existan más 0 en la matriz.

- Paso 3: Ya que tienen todo mapeado empezaran a buscar y recoger toda la basura que tienen cerca, llevándola y dejándola en la papelera el momento que lleguen a su máxima capacidad.

# Solución a nivel tecnico.

1. Importaciones: Se importan las bibliotecas necesarias. Como mesa para el uso de los agentes y modelos, se importan herramientas de `matplotlib` para la visualización, así como numpy para manejar y procesar datos.
2. Clase `Robot: Esta clase representa un robot individual en el modelo. El robot tiene propiedades como:
    - pos: posición actual en el grid.
    - trash_capacity: capacidad máxima de basura que puede llevar.
    - current_trash: cantidad actual de basura que lleva.
    - trash_bin: posición del contenedor de basura.
    - mapping: un booleano que indica si el robot está en modo mapeo.
    - target_position: una posición objetivo.
    - previous_pos: almacena la posición previa para evitar que el robot retroceda.
    - Varios otros atributos para su funcionalidad.
    - Los robots también tienen métodos para moverse, recoger basura y mapear la oficina.
3. Función `read_input': Lee un archivo que representa la disposición de la oficina. Devuelve las dimensiones de la oficina y su diseño.
4. Función `get_grid`: Obtiene el estado actual del grid del modelo.
5. El modelo Oficina: Representa el modelo de la oficina en sí. Algunas de sus características principales incluyen:
    - grid: la representación espacial de la oficina.
    - colmena: una representación numérica de la oficina.
    - schedule: un programador para decidir en qué orden se activan los agentes.
    - datacollector: recopila datos del modelo en cada paso de tiempo.
    - Se crean robots y se colocan en el grid de la oficina basándose en la disposición inicial.
6. Simulación: Se crea una instancia del modelo `Oficina` y se itera durante un máximo número iteraciones, activando los agentes y mandando sus acciones
7. Recopilación de datos: Después de la simulación, el código cuenta ciertos tipos de celdas en la "colmena".
8. Función `save_transformed_grid_to_txt`: Guarda la representación de la oficina en un archivo de texto.
9. Visualización: Se utiliza `matplotlib` para visualizar la evolución del grid a lo largo del tiempo. Se prepara una animación que muestra cómo cambia el estado del grid.

## Mecanismos de comunicación entre los agentes.

Cada robot actúa independientemente en función de su conocimiento del entorno (model.colmena y model.grid_data). Como cada robot actualiza model.colmena al mapear su entorno, otros robots que posteriormente verifiquen model.colmena estarán trabajando con la información actualizada por todos los robots que actuaron antes que ellos en esa iteración. De esta manera se comunican

entre sí ya que cada robot va a reportar al model.colmena que es lo que hay alrededor de donde pasó. Gracias a esto todos los robots saben donde están ubicadas las diferentes pilas de basura y los obstaculos. De esta forma todos los robots saben cuando ya el mapa ha sido mapeado completamente.

Para entrar en más detalle al inicializar la oficina esta matriz colmena está compuesta por un monton de 0s. Estos representan casillas que no han sido mapeadas por ningun robot. Cuando llegan a su posición inicial (utilizando el algoritmo descrito en la sección anterior) cambian su target.position a la celdas con 0 más cerca del robot. Cuando un robot mapea una zona el valor cambia dependiendo de lo que haya encontrado. Si es basura cambia el 0 por un 1 por ejemplo para que al final del mapea ya sepan donde está la basura que van a buscar. Si es simple una celda por la cual se puede caminar se cambia a un 50 para indicar que no hay nada y es un posible camino para los robots llegar a sus destinos. Si ya recogieron toda la basura de una celda se cambia del 1 al 50.

De esta forma se van comunicando las posiciones que ya limpiaron, en las que hay basura, en donde hay obstaculos. Así logran su objetivo de limpiar toda la oficina.

## Algoritmo utilizado

El algoritmo que se utiliza para que los robots se muevan hacia su `target_position` es un enfoque heurístico simple basado en la distancia Manhattan entre la posición actual del robot y la posición objetivo.

Aquí están los pasos clave:

1. Calcula la distancia Manhattan entre la posición actual del robot (`self.pos`) y la `target_position`.

2. Si la `distance_to_target` es mayor que 2 (lo que asumo es una decisión de diseño para considerar que el robot está "cerca" de su objetivo si está a una distancia Manhattan de 2 o menos), entonces procede a mover el robot en la dirección de `target_position`.

3. Obtiene los posibles pasos o movimientos que el robot puede hacer desde su posición actual:

4. De entre estos pasos posibles, elige el siguiente paso que minimice la distancia Manhattan a la `target_position`:

5. Mueve al robot a `next_position`:

El algoritmo básicamente hace que el robot siempre se mueva en la dirección que reduzca la distancia a su objetivo. Esto lo usamos para basicamente todo. Cuando van su posición inicial para empezar el mapeo su target_position son las esquinas y el centro. A la hora del mapeo buscan los 0 en la matriz colmena. A la hora de recoger la basura buscan los 1 en la matriz colmena también e igual para ir directo a la papelera cuando están llenos.