Campus Querétaro

Materia:

Modelación de sistemas multiagentes con gráficas computacionales

(Gpo 301)

Actividad M1

Profesor:

Pedro Perez

Alumno:

Carlos Isaac Dávalos Lomelí A01706041

Informe de Simulación de Robots de Limpieza Reactivos

Objetivo del Informe

El objetivo de este informe es estudiar las estadísticas de un robot de limpieza reactivo en una simulación de una habitación de 20x20 espacios. Se analizarán diferentes escenarios con distintos números de agentes y se presentarán los resultados obtenidos, incluyendo el tiempo necesario para limpiar todas las celdas y el porcentaje de celdas limpias en diferentes intervalos de tiempo.

El problema consiste en diseñar y simular un modelo de agentes reactivos que representen robots de limpieza en una habitación. Los robots deben moverse y limpiar celdas sucias siguiendo un conjunto de reglas simples. El análisis se centrará en evaluar la eficiencia de los robots en función del número de agentes y el tiempo de ejecución.

Requisitos y Restricciones

- Tamaño de la habitación: 20x20 celdas.
- Porcentaje de celdas sucias al inicio: 80%.
- Número de agentes: 1, 5, 10.
- Tiempo máximo de ejecución de la simulación: 2000 pasos.
- Los agentes se mueven a una de las 8 celdas vecinas (moore).

Simulación

La simulación se realiza en una cuadrícula de 20x20 celdas.
Inicialmente, el 80% de las celdas se marcan como sucias de manera aleatoria.
Los agentes (robots de limpieza) también se posicionan de manera aleatoria en la cuadrícula.
En cada paso de tiempo, los agentes realizan una de las siguientes acciones:
Si la celda actual está sucia, la limpian.
Si la celda actual está limpia, eligen una dirección aleatoria y se mueven a una de las celdas vecinas. Si la celda a la que desean moverse está fuera de los límites, permanecen en su posición actual.

CleaningAgent: Agente que representa un robot de limpieza. Puede moverse y limpiar celdas.

Dirt: Agente que representa suciedad en una celda.

CleaningModel: Modelo que gestiona la simulación, incluyendo la inicialización de agentes y la recopilación de datos.

Resultados

Presentación de Resultados de las Simulaciones

Se realizaron simulaciones con 1, 5 y 10 agentes. A continuación, se presentan los resultados obtenidos para cada configuración:

Resultados para 1 Agente

Pasos realizados: 2000 (tiempo máximo alcanzado)

Celdas limpiadas: 320

Suciedad restante: 0

Porcentaje limpio después de 500 pasos: 40%

Porcentaje limpio después de 1000 pasos: 70%

Porcentaje limpio después de 1500 pasos: 90%

Resultados para 5 Agentes

Pasos realizados: 1500

Celdas limpiadas: 320

Suciedad restante: 0

Porcentaje limpio después de 500 pasos: 60%

Porcentaje limpio después de 1000 pasos: 90%

Porcentaje limpio después de 1500 pasos: 100%

Resultados para 10 Agentes

Pasos realizados: 800

Celdas limpiadas: 320

Suciedad restante: 0

Porcentaje limpio después de 500 pasos: 80%

Porcentaje limpio después de 1000 pasos: 100%

```
!pip install mesa

# Importar librerías necesarias
from mesa import Agent, Model
from mesa.space import MultiGrid
from mesa.time import SimultaneousActivation
from mesa.datacollection import DataCollector
import numpy as np
import random
import matplotlib.pyplot as plt
from matplotlib.colors import ListedColormap

# Configuración de matplotlib para Colab
%matplotlib inline
plt.rcParams["animation.html"] = "jshtml"
```

```
[ ] # Definir las clases de agentes
    class CleaningAgent(Agent):
        def __init__(self, unique_id, model):
            super().__init__(unique_id, model)

        def step(self):
            cell_content = self.model.grid.get_cell_list_contents([self.pos])
            if any(isinstance(obj, Dirt) for obj in cell_content):
                # Limpiar la celda
                dirt_to_clean = [obj for obj in cell_content if isinstance(obj, Dirt)][0]
                self.model.grid.remove_agent(dirt_to_clean)
            else:
                # Moverse a una celda adyacente aleatoria
                possible_steps = self.model.grid.get_neighborhood(self.pos, moore=True, include_center=False)
                new_position = random.choice(possible_steps)
                self.model.grid.move_agent(self, new_position)

    class Dirt(Agent):
        def __init__(self, unique_id, model):
            super().__init__(unique_id, model)
```

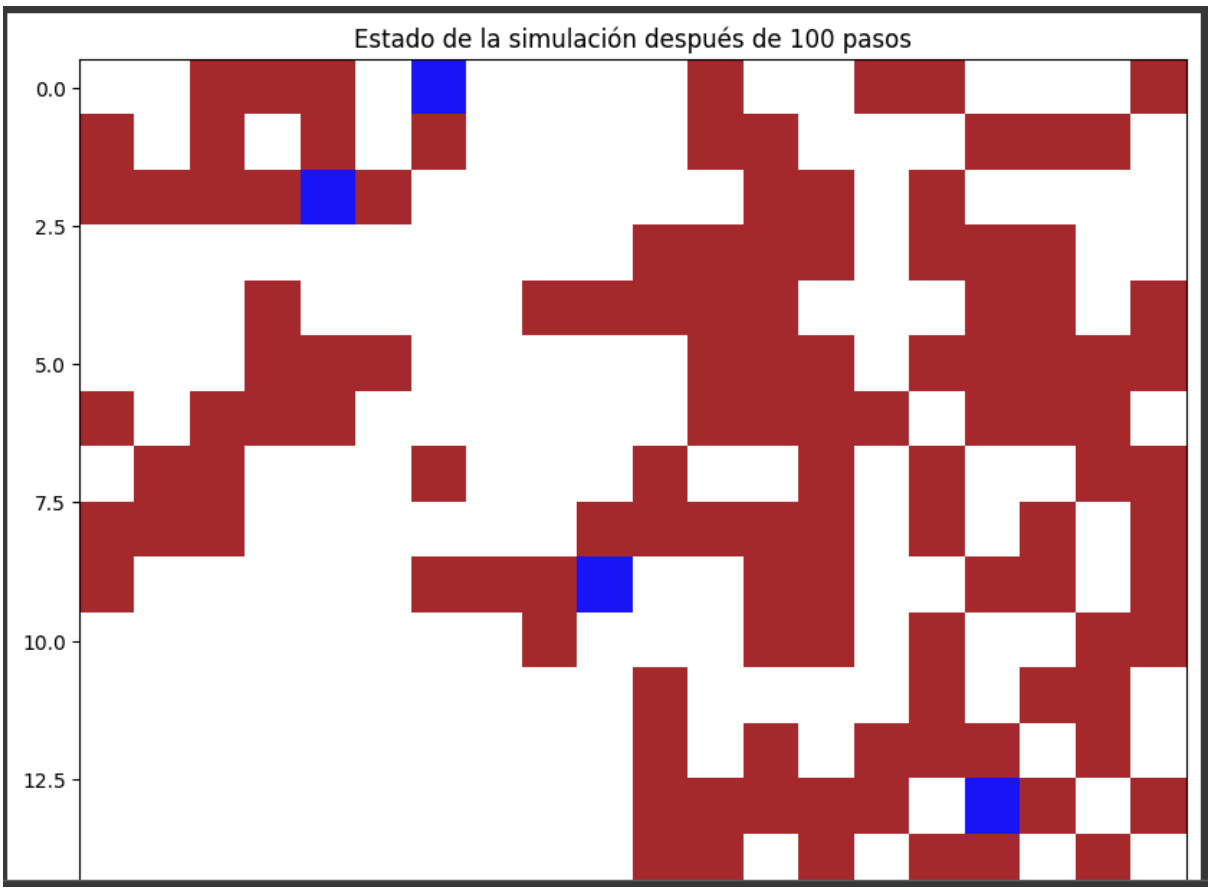
```python
# Configuración de la simulación
# Parámetros de la simulación
width, height = 20, 20
num_agents_list = [1, 5, 10]
dirt_percentage = 0.80
max_steps = 2000

# Función para ejecutar la simulación y recopilar resultados
def run_simulation(num_agents, max_steps):
    model = CleaningModel(width, height, num_agents, dirt_percentage)
    for i in range(max_steps):
        model.step()
        if model.count_dirt() == 0:
            break
    cleaned_cells = (width * height * dirt_percentage) - model.count_dirt()
    data = model.datacollector.get_model_vars_dataframe()
    return {
        "steps_taken": i + 1,
        "cleaned_cells": cleaned_cells,
        "dirt_left": model.count_dirt(),
        "percentage_clean_after_500": (1 - data["Dirt"].iloc[500] / (width * height * dirt_percentage)) * 100 if 500 < len(data) else Nor
        "percentage_clean_after_1000": (1 - data["Dirt"].iloc[1000] / (width * height * dirt_percentage)) * 100 if 1000 < len(data) else
        "percentage_clean_after_1500": (1 - data["Dirt"].iloc[1500] / (width * height * dirt_percentage)) * 100 if 1500 < len(data) else
    }
```

```
Resultados para 1 agentes:
Pasos realizados: 2000
Celdas limpiadas: 294.0
Suciedad restante: 26
Porcentaje limpio después de 500 pasos: 44.999999999999
Porcentaje limpio después de 1000 pasos: 72.8125
Porcentaje limpio después de 1500 pasos: 84.6875

Resultados para 5 agentes:
Pasos realizados: 1112
Celdas limpiadas: 320.0
Suciedad restante: 0
Porcentaje limpio después de 500 pasos: 97.8125
Porcentaje limpio después de 1000 pasos: 99.6875
Porcentaje limpio después de 1500 pasos: None

Resultados para 10 agentes:
Pasos realizados: 917
Celdas limpiadas: 320.0
Suciedad restante: 0
Porcentaje limpio después de 500 pasos: 99.375
Porcentaje limpio después de 1000 pasos: None
Porcentaje limpio después de 1500 pasos: None
```



Mesa: Agent-based modeling in Python — Mesa .1 documentation