Tecnológico de Monterrey

M1. Actividad

Luis Gabriel Delfín Paulín

A01701482

28/05/2024

Descripción

Se busca hacer una simulación que contenga lo siguiente:

- Inicializa las celdas sucias (ubicaciones aleatorias).
- Todos los agentes empiezan en localidades aleatorios.
- En cada paso de tiempo:
    - Si la celda está sucia, entonces aspira.
    - Si la celda está limpia, el agente elije una dirección aleatoria para moverse (unas de las 8 celdas vecinas) y elije la acción de movimiento (si no puede moverse allí, permanecerá en la misma celda).
- Se ejecuta el tiempo máximo establecido.

De igual manera es indispensable recopilar la siguiente información durante la ejecución:

- Tiempo necesario hasta que todas las celdas estén limpias (o se haya llegado al tiempo máximo).
- Porcentaje de celdas limpias después de 500, 1000 y 1500 pasos de simulación.
- ¿Cuál es la cantidad mínima de agentes para lograr que el espacio esté limpio en menos de 1000 pasos?

Código explicación

Este código primero corre el modelo hasta que todas las celdas estén limpias, después imprime los resultados y finalmente crea y muestra la animación.

Clase Agente

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

    def step(self):
        if self.model.room[self.pos] == 1:
            # Aspira la celda
            self.model.room[self.pos] = 0
        else:
            # Mueve a una celda vecina 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)
```

Se escogió una lógica muy simple para la clase del agente, donde definimos que 1 es una celda sucia y 0 es una celda limpia. Igual se busca hacer un cambio de posición aleatoria, que es lo que se busca, sin buscarle mucha lógica, ya que es fundamental que el agente no sepa qué posiciones están sucias.

Clase Modelo

```
class CleaningModel(Model):
    def __init__(self, num_agents, width, height, dirty_percentage):
        super().__init__()
        self.num_agents = num_agents
        self.grid = MultiGrid(width, height, False)
        self.schedule = RandomActivation(self)
        self.current_step = 0
        self.room = {}

        # Inicializa la habitación con celdas sucias
        num_dirty_cells = int(width * height * dirty_percentage)
        dirty_cells = random.sample([(x, y) for x in range(width) for y in range(height)], num_dirty_cells)
        for cell in dirty_cells:
            self.room[cell] = 1
        for cell in [(x, y) for x in range(width) for y in range(height)]:
            if cell not in self.room:
                self.room[cell] = 0
```

```
        # Inicializa los agentes
        for i in range(self.num_agents):
            agent = VacuumAgent(i, self)
            self.schedule.add(agent)
            x = random.randrange(self.grid.width)
            y = random.randrange(self.grid.height)
            self.grid.place_agent(agent, (x, y))

    def step(self):
        self.schedule.step()
        self.current_step += 1
        if self.calculate_clean_percentage() == 100:
            self.running = False

    def calculate_clean_percentage(self):
        clean_cells = sum(1 for cell in self.room.values() if cell == 0)
        total_cells = self.grid.width * self.grid.height
        clean_percentage = (clean_cells / total_cells) * 100
        return clean_percentage
```

En esta parte del código donde inicializamos la clase modelo, lo que se busca es crear los atributos de nuestra simulación, así como las celdas sucias, se añaden los agentes y contamos los “steps” de la simulación. Aquí igual es importante decir que si llegáramos a tener el 100% de las celdas limpias, la simulación se detiene.

Crear animación

```
# Función para crear la animación
def animate_model(model):
    fig, ax = plt.subplots(figsize=(6, 6))

    def update(frame):
        if not model.running:
            return  # Detener la animación si el modelo ha dejado de correr

        ax.clear()
        grid = np.zeros((model.grid.width, model.grid.height))
        for cell in model.room:
            grid[cell] = model.room[cell]
        ax.imshow(grid, cmap='gray')

        ax.set_title(f'Step: {model.current_step}')
        ax.set_xticks([])
        ax.set_yticks([])
        model.step()

    anim = animation.FuncAnimation(fig, update, frames=None, repeat=False)
    plt.close()
    return anim
```

Aquí tomamos nuestro modelo para empezar a animar nuestra simulación, donde creamos una cuadrícula de ceros, que es donde definimos que 1 es que está sucio y 0 limpio.

Parámetros

```
# Parámetros de la simulación
width = 20
height = 20
dirty_percentage = 0.80
num_agents_list = [1, 5, 10]
check_steps = [500, 1000, 1500]
```

Aquí es donde nosotros metemos los parámetros que le queremos poner a la simulación, ya sea un mayor espacio, más o menos celdas sucias, el número de agentes y los steps que queremos que corra la simulación.

Ejecución de simulación y animación

```
# Ejecutar la simulación y animar
animations = []  # Lista para mantener las animaciones en memoria
for num_agents in num_agents_list:
    model = CleaningModel(num_agents, width, height, dirty_percentage)

    # Variable para mantener el estado de limpieza de cada paso
    clean_percentage_by_step = {}

    # Ejecutar todos los pasos del modelo
    while model.running:
        model.step()

        # Verificar si el paso actual coincide con los pasos a verificar
        if model.current_step in check_steps:
            clean_percentage = model.calculate_clean_percentage()
            clean_percentage_by_step[model.current_step] = clean_percentage
            print(f"Porcentaje de celdas limpias en el paso {model.current_step} con {num_agents} agentes: {clean_percentage:.2f}%")
```

```
    # Mostrar el porcentaje en los pasos especificados aunque se haya alcanzado el 100% antes
    for step in check_steps:
        if step not in clean_percentage_by_step:
            clean_percentage = model.calculate_clean_percentage()
            clean_percentage_by_step[step] = clean_percentage
            print(f"Porcentaje de celdas limpias en el paso {step} con {num_agents} agentes: {clean_percentage:.2f}%")

    # Verificar si se alcanzó el 100% de celdas limpias
    if model.calculate_clean_percentage() == 100:
        print(f"Paso en el que se alcanzó el 100% de celdas limpias con {num_agents} agentes: {model.current_step}")

    anim = animate_model(model)
    animations.append(anim)  # Mantener la animación en memoria
```

Finalmente mostramos los resultados de nuestra simulación y corremos la animación. Al ser una simulación con steps “infinitos” hasta que nuestros agentes limpien todo, la animación se tarda en crear.

Simulación

```
Porcentaje de celdas limpias en el paso 500 con 1 agentes: 48.50%
Porcentaje de celdas limpias en el paso 1000 con 1 agentes: 68.50%
Porcentaje de celdas limpias en el paso 1500 con 1 agentes: 82.50%
Paso en el que se alcanzó el 100% de celdas limpias con 1 agentes: 6208
Porcentaje de celdas limpias en el paso 500 con 5 agentes: 92.75%
Porcentaje de celdas limpias en el paso 1000 con 5 agentes: 100.00%
Porcentaje de celdas limpias en el paso 1500 con 5 agentes: 100.00%
Paso en el que se alcanzó el 100% de celdas limpias con 5 agentes: 959
Porcentaje de celdas limpias en el paso 500 con 10 agentes: 99.75%
Porcentaje de celdas limpias en el paso 1000 con 10 agentes: 100.00%
Porcentaje de celdas limpias en el paso 1500 con 10 agentes: 100.00%
```

Concluimos con esta simulación donde contestamos a las preguntas que se hicieron al inicio que fueron:

- **Tiempo necesario hasta que todas las celdas estén limpias (o se haya llegado al tiempo máximo).**

En la simulación se puede observar los steps que tomaron (1, 5, 10) agentes para limpiar todas las celdas.

- **Porcentaje de celdas limpias después de 500, 1000 y 1500 pasos de simulación.**
  En la simulación se puede observar cuántas celdas limpiaron después de 500, 1000 y 1500 pasos.

- **¿Cuál es la cantidad mínima de agentes para lograr que el espacio esté limpio en menos de 1000 pasos?**
  Cada simulación varía, pero podemos observar que con 5 agentes es posible que llegue a haber un espacio 100% limpio.

Repositorio Github
https://github.com/Gabriel-Delfin/Aspiradora