Simulación de enjambre de abejas

28 de mayo de 2025

Facultad de Estadística e Informática

Docente: Fred Torres Cruz
Repositorio GitHub: Simulación Abejas
Nombre: Adelmi Cordova Apaza
Código: 230850

Estrategia de Modelado

Se modelan los siguientes tipos de agentes:

- **Abeja:** Se desplaza en el entorno en búsqueda de flores. Extrae néctar y lo lleva a la colmena.
- **Flor:** Contiene una cantidad limitada de néctar. Se encuentra distribuida aleatoriamente.
- **Colmena:** Almacena el néctar recolectado por las abejas.

Las abejas recolectan néctar y regresan a la colmena. Si no hay flores, las abejas regresan sin recolectar.

Código Fuente

```python
from mesa import Agent, Model
from mesa.space import MultiGrid
from mesa.time import RandomActivation
import random

class Abeja(Agent):
    def __init__(self, unique_id, model):
        super().__init__(unique_id, model)
        self.nectar = 0

    def step(self):
        self.mover()
        self.recolectar()
        self.entregar()

    def mover(self):
        posibles = self.model.grid.get_neighborhood(
            self.pos, moore=True, include_center=False)
        nueva_pos = random.choice(posibles)
        self.model.grid.move_agent(self, nueva_pos)

    def recolectar(self):
        celda = self.model.grid.get_cell_list_contents([self.pos])
        for obj in celda:
            if isinstance(obj, Flor) and obj.nectar > 0:
                self.nectar += 1
                obj.nectar -= 1
                break

    def entregar(self):
        celda = self.model.grid.get_cell_list_contents([self.pos])
        for obj in celda:
            if isinstance(obj, Colmena):
                obj.nectar += self.nectar
                self.nectar = 0
                break

class Flor(Agent):
    def __init__(self, unique_id, model):
        super().__init__(unique_id, model)
        self.nectar = random.randint(1, 3)

    def step(self):
        pass

class Colmena(Agent):
    def __init__(self, unique_id, model):
        super().__init__(unique_id, model)
        self.nectar = 0

    def step(self):
        pass
```

```
53
54 class ModeloAbejas(Model):
55     def __init__(self, ancho, alto, n_abejas, n_flores):
56         self.grid = MultiGrid(ancho, alto, torus=False)
57         self.schedule = RandomActivation(self)
58
59         # Crear colmena
60         colmena = Colmena(0, self)
61         self.schedule.add(colmena)
62         self.grid.place_agent(colmena, (ancho // 2, alto // 2))
63
64         # Crear abejas
65         for i in range(1, n_abejas + 1):
66             abeja = Abeja(i, self)
67             self.schedule.add(abeja)
68             x = random.randint(0, ancho - 1)
69             y = random.randint(0, alto - 1)
70             self.grid.place_agent(abeja, (x, y))
71
72         # Crear flores
73         for i in range(n_abejas + 1, n_abejas + n_flores + 1):
74             flor = Flor(i, self)
75             self.schedule.add(flor)
76             x = random.randint(0, ancho - 1)
77             y = random.randint(0, alto - 1)
78             self.grid.place_agent(flor, (x, y))
79
80     def step(self):
81         self.schedule.step()
```

Visualización de Resultados

El modelo permite observar cómo las abejas buscan flores, recolectan néctar y lo depositan en la colmena. El comportamiento colectivo se puede analizar observando el flujo de néctar en el sistema.

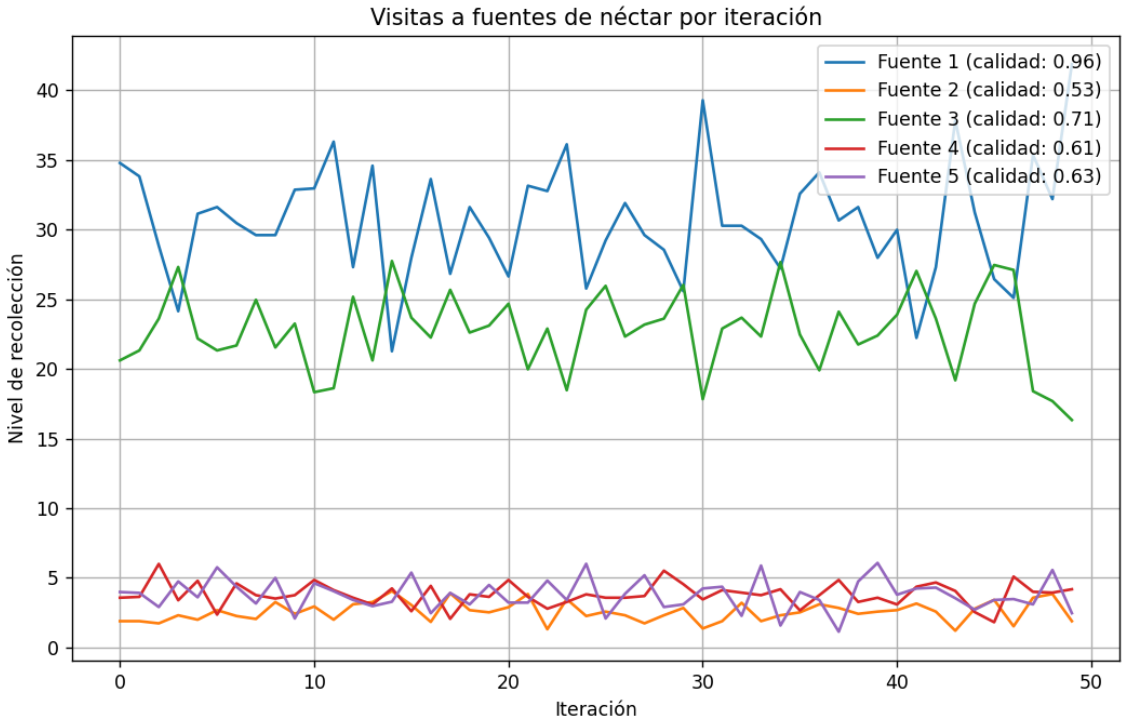

Figura 1: Resultado del código fuente

```
✓ Fuente más visitada: Fuente 4
              mean       std        total
Fuente 1  22.779925  2.556151  1138.996229
Fuente 2   2.820132  0.825607   141.006610
Fuente 3   2.852578  0.908496   142.628917
Fuente 4  31.054124  3.438644  1552.706203
Fuente 5   3.434043  0.993852   171.702168
PS C:\Users\HP> & C:/Users/HP/AppData/Local/Microsoft/WindowsApps/pyt
✓ 100 abejas asignadas: 30 exploradoras, 50 obreras, 20 seguidoras
✓ Se generaron 5 fuentes de néctar con calidad aleatoria
✓ Simulación completada en 50 iteraciones
```

Aplicaciones del Modelo

Este modelo bioinspirado tiene múltiples aplicaciones prácticas:

- **Optimización de Rutas (Logística):** El comportamiento de búsqueda de las abejas puede utilizarse en algoritmos como Bee Colony Optimization.
- **Sistemas Distribuidos e Inteligencia Artificial:** Diseño de enjambres de drones o robots colaborativos que actúan localmente.
- **Economía y Toma de Decisiones Colectivas:** Modelos de mercado con múltiples agentes basados en señales locales.
- **Ecología y Conservación:** Modela el comportamiento de polinizadores y su impacto ecológico.

Conclusiones

- Las abejas logran colectivamente enfocarse en las fuentes de mejor calidad.
- A pesar de la aleatoriedad individual, el grupo converge hacia soluciones eficientes.
- Este comportamiento puede ser aplicado en problemas de optimización, logística, inteligencia artificial, entre otros.

Repositorio en GitHub

Puedes acceder al código completo en GitHub:
`https://github.com/Adelmi195/Simulacion-abejas/blob/main/modelo_abejas_F.ipynb`