1. Actividad N°9

Escoja usted un sistema bioinspirado y replicar el notebook con un nuevo ejemplo.

FORRAGEO DE HORMIGAS

Nombre Completo: David León Callohuanca Condori
Curso: Estadística Computacional
Carrera: Ingeniería Estadística e informática
Universidad: Universidad Nacional del Altiplano
Fecha: 27 de Mayo 2025

1. **Modelo ACO:**
   Es un algoritmo bioinspirado basado en el comportamiento colectivo de las hormigas al buscar alimento, donde agentes individuales (hormigas) exploran un entorno en forma de rejilla, dejando feromonas sobre las rutas más eficientes entre el nido y las fuentes de comida. Estas feromonas sirven como una memoria colectiva indirecta: otras hormigas tienden a seguir y reforzar las rutas con mayor concentración, lo que conduce a la emergencia de caminos óptimos. En el ejemplo simulado, las hormigas actúan de forma autónoma, perciben niveles locales de feromona, toman decisiones probabilísticas y contribuyen, sin coordinación central, a la construcción de rutas eficientes entre el nido y la comida.

   - **Pregunta de investigación:** ¿Cómo optimiza la recolección de recursos un sistema multiagente basado en el comportamiento de forrajeo de hormigas?

2. **Configuración del modelo:**

   a) **Paso 1: Agentes**
   Tipo de agente: Hormigas
   Cantidad inicial: 100 hormigas (ajustable)

   Atributos de cada agente:

   - **estado:** buscando y regresando
   - **posición:** coordenadas (x, y) en la rejillas
   - **memoria_ruta:** lista de celdas recorridas al buscar comida
   - **carga:** booleana (si lleva comida o no)

   Capacidades:

   - Percibir feromonas en celdas vecinas
   - Detectar comida en celdas adyacentes
   - Moverse a una celda vecina
   - Dejar feromonas (si está regresando)
   - Cambiar de estado según eventos

   Función de cambio de estado del agente:

   - Cuando una hormiga encuentra comida:
     Si `comida_en_celda(x, y)` = `True` $\Rightarrow$ `estado` $\leftarrow$ `regresando`
   - Cuando llega al nido con comida:
     Si `posicion` = `nido` $\wedge$ `estado` = `regresando` $\Rightarrow$ `estado` $\leftarrow$ `buscando`

   Condición de percepción local:

   - Una hormiga solo puede "ver" las feromonas o comida en sus celdas vecinas inmediatas (por ejemplo, en un radio de 1): `vecinos(i, j)` = {`(i', j')` | |`i - i'`| $\leq 1 \wedge$ |`j - j'`| ≤ 1 }

b) **Paso 2: Entorno**
Tipo de entorno: Rejilla bidimensional

Elementos clave del entorno:

- **Nido:** ubicación fija
- **Comida:** varias fuentes colocadas aleatoriamente
- **Feromonas:** cada celda almacena un nivel de feromona, que se evapora con el tiempo

Características:

- Espacio discreto (cada celda puede ser ocupada por agentes y contener niveles de feromona)
- Visión local: los agentes solo perciben celdas adyacentes
- Dinámico: los niveles de feromona cambian, y la comida puede agotarse

El entorno en este modelo se representa con una rejilla bidimensional discreta en la que cada celda contiene:

- Nivel de feromona τ_{ij}
- Presencia de comida: booleana o cantidad
- Tipo de celda: nido, comida o libre

Evaporación en el entorno (como ya se mostró):

- $\tau_{ij} \leftarrow (1-\rho)\cdot \tau_{ij}$
  Esto es una actualización del entorno, que ocurre en cada paso del tiempo.

c) **Paso 3: Comportamiento**
Ciclo de vida de una hormiga: Buscar comida:

- Se mueve hacia una celda vecina con mayor feromona (probabilístico)
- Si no hay feromonas, se mueve aleatoriamente
- Si encuentra comida, cambia a estado 'regresando' y guarda la ruta recorrida

Regresar al nido:

- Sigue la ruta inversa guardada
- Deja feromonas en cada celda que atraviesa (la cantidad puede depender de la calidad o distancia del camino)
- Al llegar al nido, deja la comida y cambia a estado 'buscando'

Evaporación de feromonas:

- En cada paso de simulación, todas las celdas reducen su nivel de feromona un porcentaje (simulando evaporación)

Formulas:

1) **Buscar comida (exploración):**
   Cada hormiga, estando en el estado `"buscando"`, evalúa sus celdas vecinas y decide a cuál moverse basándose en dos factores:
   - Nivel de feromona (τ_{ij}): qué tan fuerte es la señal de feromona en la celda vecina.
   - Heurística local (η_{ij}): por ejemplo, cercanía aparente al objetivo (opcional).

   La probabilidad de elegir la celda j desde la celda actual i se calcula con:

$$P_{ij} = \frac{[\tau_{ij}]^\alpha \cdot [\eta_{ij}]^\beta}{\sum_{k \in N_i} [\tau_{ik}]^\alpha \cdot [\eta_{ik}]^\beta}$$

   Donde:
   - P_{ij} es la probabilidad de que la hormiga se mueva de i a j.
   - α controla la importancia de la feromona.
   - β controla la importancia de la heurística (si no se usa, $\beta = 0$).
   - N_i es el conjunto de celdas vecinas accesibles desde i.

2) **Encontrar comida y regresar (explotación):**
Al encontrar comida, la hormiga cambia al estado 'regresando'.
Sigue el camino inverso almacenado, depositando feromona en cada celda visitada.
La cantidad de feromona que una hormiga deja en cada celda se define como:

$$\Delta\tau_{ij} = \frac{Q}{L}$$

Donde:

- Q es una constante que representa la cantidad total de feromona que puede depositar.
- L es la longitud del camino recorrido (puede ser tiempo o número de pasos).

3) **Evaporación de feromonas:**
En cada celda, la feromona se disminuye con el tiempo para evitar que las rutas antiguas dominen para siempre:

$$\tau_{ij} \leftarrow (1-\rho)\cdot\tau_{ij}$$

Donde:

- ρ es la tasa de evaporación $(0 < \rho < 1)$.

Valores típicos: $\rho = 0{,}1$ o $\rho = 0{,}5$.

3. **CODIGO:**

```
import numpy as np
import matplotlib.pyplot as plt
import matplotlib.colors as mcolors
import matplotlib.animation as animation
import random


GRID_SIZE = 50
NUM_ANTS = 100
NUM_STEPS = 200
FOOD_SOURCES = 5
EVAPORATION_RATE = 0.05
PHEROMONE_DEPOSIT = 1.0


pheromone_grid = np.zeros((GRID_SIZE, GRID_SIZE))
food_grid = np.zeros((GRID_SIZE, GRID_SIZE))
nest_position = (GRID_SIZE // 2, GRID_SIZE // 2)
total_food_start = 0


for _ in range(FOOD_SOURCES):
    x, y = random.randint(0, GRID_SIZE-1), random.randint(0, GRID_SIZE-1)
    food_amount = 50
    food_grid[x, y] = food_amount
    total_food_start += food_amount


class Ant:
    def __init__(self, nest):
        self.x, self.y = nest
        self.state = "searching"
        self.path = []

    def get_neighbors(self):
        deltas = [(-1,0), (1,0), (0,-1), (0,1)]
        return [(self.x + dx, self.y + dy) for dx, dy in deltas
                if 0 <= self.x + dx < GRID_SIZE and 0 <= self.y + dy <
                    GRID_SIZE]

```

```
    def move(self, pheromone, food, nest):
        neighbors = self.get_neighbors()

        if self.state == "searching":
            weights = [(pheromone[nx, ny] + 1e-6) for nx, ny in neighbors]
            total = sum(weights)
            probs = [w / total for w in weights]
            next_x, next_y = random.choices(neighbors, weights=probs)[0]
            self.path.append((self.x, self.y))
            self.x, self.y = next_x, next_y

            if food[self.x, self.y] > 0:
                food[self.x, self.y] -= 1
                self.state = "returning"

        elif self.state == "returning":
            if self.path:
                pheromone[self.x, self.y] += PHEROMONE_DEPOSIT
                self.x, self.y = self.path.pop()
            if (self.x, self.y) == nest:
                self.state = "searching"
                self.path = []


ants = [Ant(nest_position) for _ in range(NUM_ANTS)]


food_collected_over_time = []


fig, (ax1, ax2) = plt.subplots(1, 2, figsize=(12, 6))
cmap = mcolors.LinearSegmentedColormap.from_list("pheromone_cmap", ["white"
    , "red"])
ant_color = "black"

def update(frame):
    ax1.clear()
    ax2.clear()


    for ant in ants:
        ant.move(pheromone_grid, food_grid, nest_position)
    pheromone_grid[:] *= (1 - EVAPORATION_RATE)


    total_food_now = food_grid.sum()
    collected = total_food_start - total_food_now
    food_collected_over_time.append(collected)


    ax1.imshow(pheromone_grid, cmap=cmap)
    ax1.set_title(f"Paso {frame} - Feromonas")
    ax1.scatter(*nest_position[::-1], color='blue', s=40, label='Nido')


    xs = [ant.y for ant in ants]
    ys = [ant.x for ant in ants]
    ax1.scatter(xs, ys, color=ant_color, s=5, label='Hormigas')
    ax1.legend(loc='upper right')
    ax1.set_xticks([])
    ax1.set_yticks([])


```

```
    ax2.plot(food_collected_over_time, color='green')
    ax2.set_title("Comida recolectada a lo largo del tiempo")
    ax2.set_xlabel("Paso de simulacion")
    ax2.set_ylabel("Total acumulado")


ani = animation.FuncAnimation(fig, update, frames=NUM_STEPS, repeat=False,
    interval=100)
plt.tight_layout()
plt.show()
```

Listing 1: Simulación de hormigas

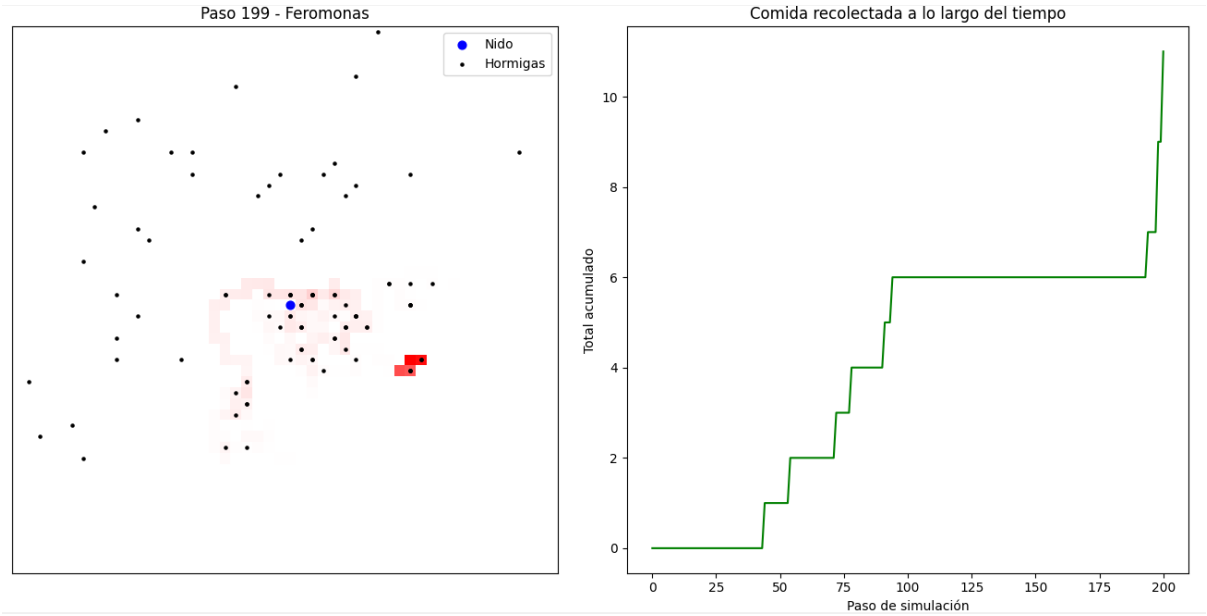

Figura 1: Ejemplo de simulación de colonia de hormigas

GIT HUB:`https://github.com/austraraptor/Agentes.git`