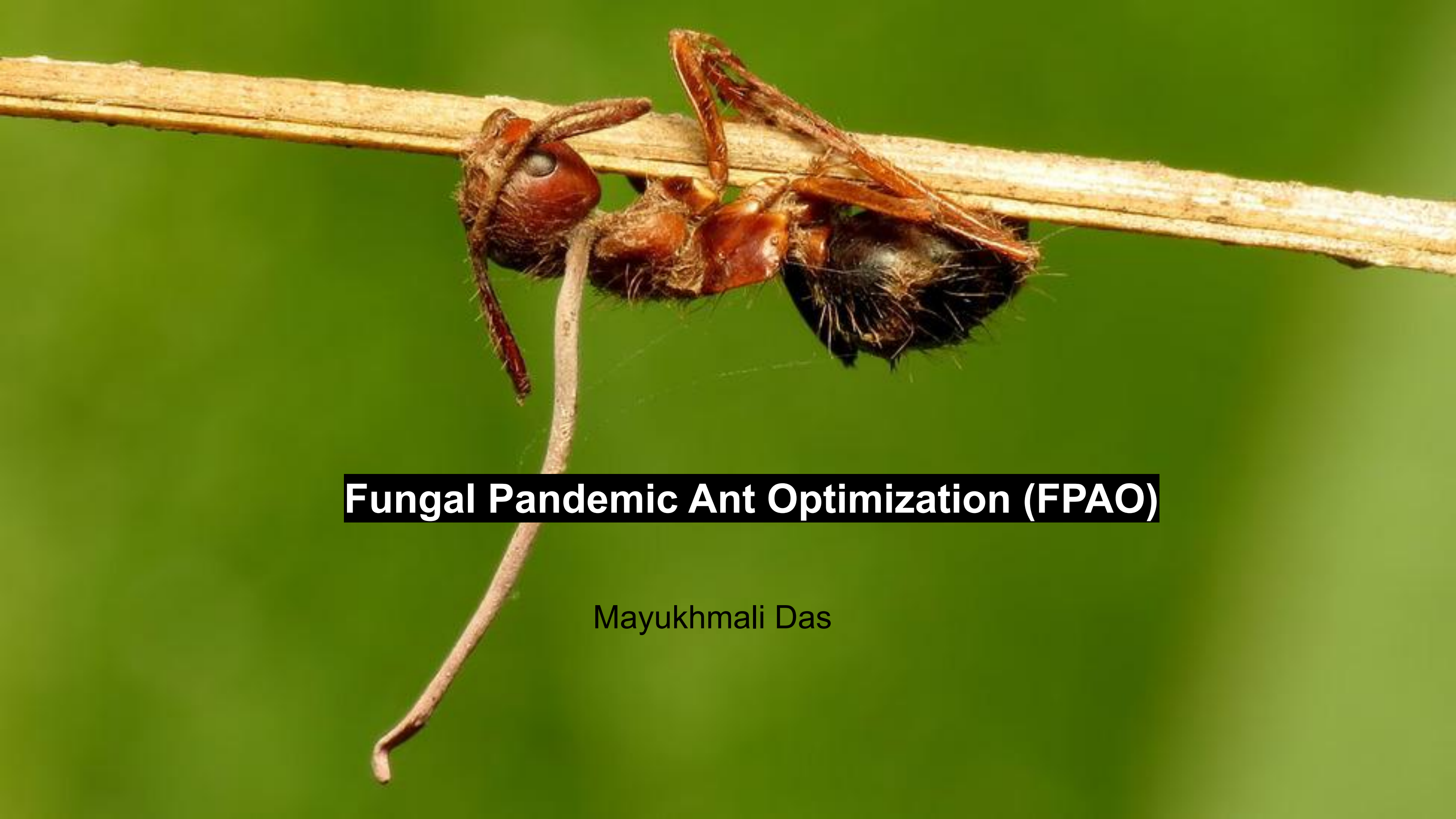


Fungal Pandemic Ant Optimization (FPAO)

Mayukhmali Das

The parasitic fungus that turns ants into "zombies" has been a subject of scientific fascination, but it's unlikely to affect humans.

However, the premise of a new TV show based on a video game called **The Last of Us** imagines a scenario where climate change causes a fungus to take over and turn humans into zombie-like creatures.

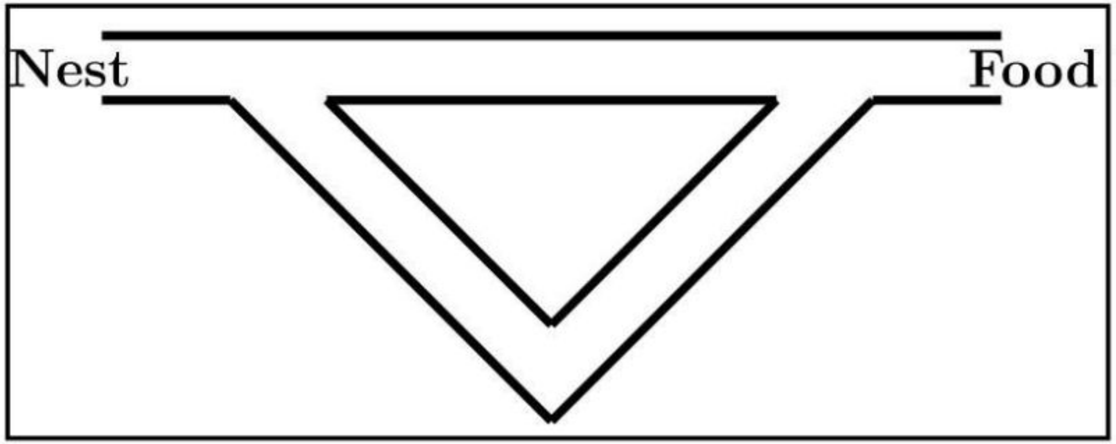
We have taken inspiration from the TV Show for our presentation

The cordyceps fungus, also known as the zombie-ant fungus, is a parasite that infects insects like ants. It drains the host of nutrients and fills its body with spores for reproduction. The fungus manipulates the insect's behavior and forces it to climb to a higher location before expelling the spores, infecting other nearby insects. This process enables the fungus to spread and infect more hosts, similar to how other parasites operate.

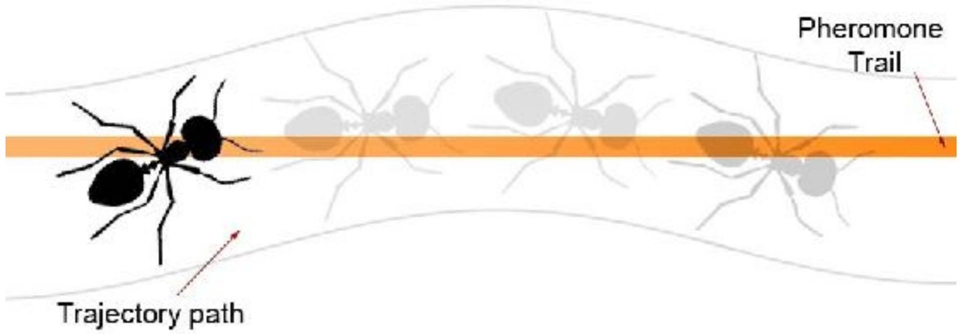
Once worker ants find the infected; they are quickly taken away and dumped far away from the colony.

We will be using **ants affected by such a fungal pandemic** to create an **optimization algorithm**.

We will be using the ***proposed algorithm*** to find the **shortest path between two nodes.**



We have two paths L1 and L2 where L2>L1



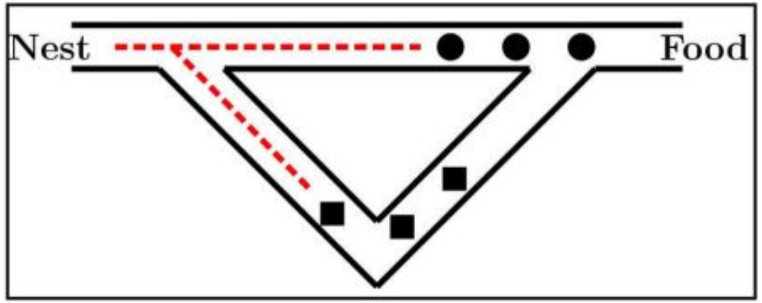
Real ants leave pheromones on the paths they travel, which we will be modelling using artificial **pheromone values (τi)** for each available path.

Each ant will choose a path whose probability is given the the formula :

$$\mathbf{p}_i = \frac{\tau_i}{\tau_1 + \tau_2}$$

Pheromone values of the paths

if pheromone of first path is greater than the second, the probability of choosing L1 is higher, and vice versa



Now the pheromone is assigned to a particular path when the ant returns.
The pheromone value of a path is updated as follows

$$\tau_i \leftarrow \tau_i + \frac{Q}{l_i}$$

where the positive constant Q is a parameter of the model. In other words, the amount of artificial pheromone that is added depends on the length of the chosen path: the shorter the path, the higher the amount of added pheromone

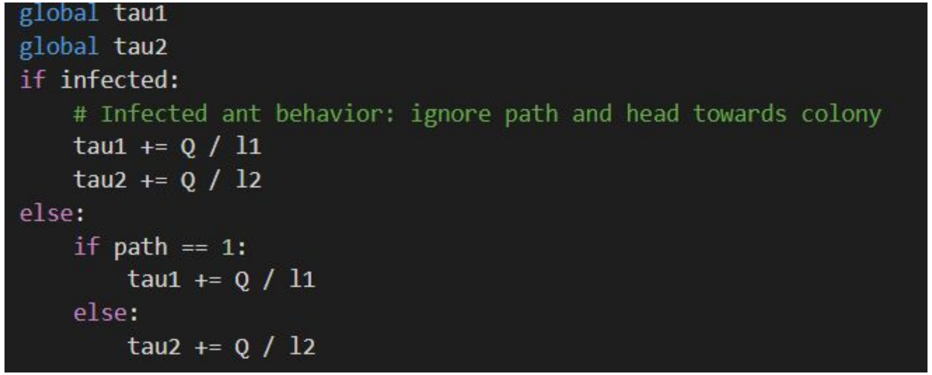
Distance travelled

We also are trying to simulate the natural process of pheromone evaporation by gradually reducing the pheromone value on each path over time. This allows the model to adapt to changing conditions and avoid outdated paths.

$$\tau_i \leftarrow (1 - \rho) \cdot \tau_i$$ Pheromone Evaporation formulae

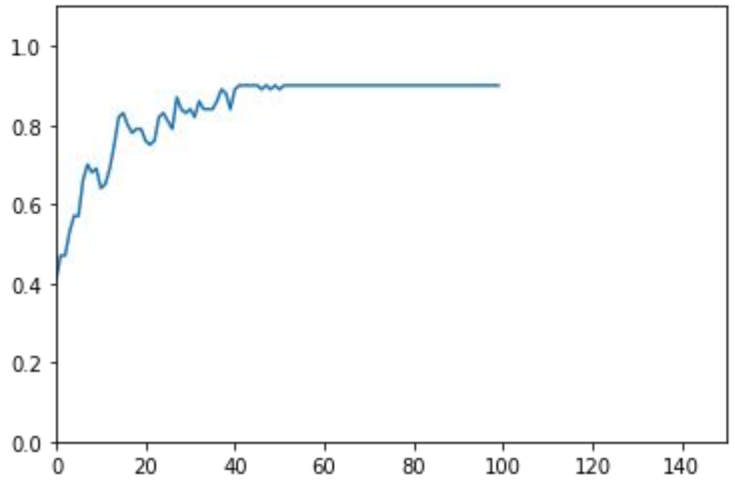
Now we are introducing the **Cordyceps fungus** to this mix. Each ant has a **probability of say 1 percent** to be affected by the cordyceps fungus.

Now at each iteration of the algorithm, it will be checked whether the ants are infected or not. **If infected; the ant will be made to ignore the path and head straight towards colony.** A small code snippet is shown below:

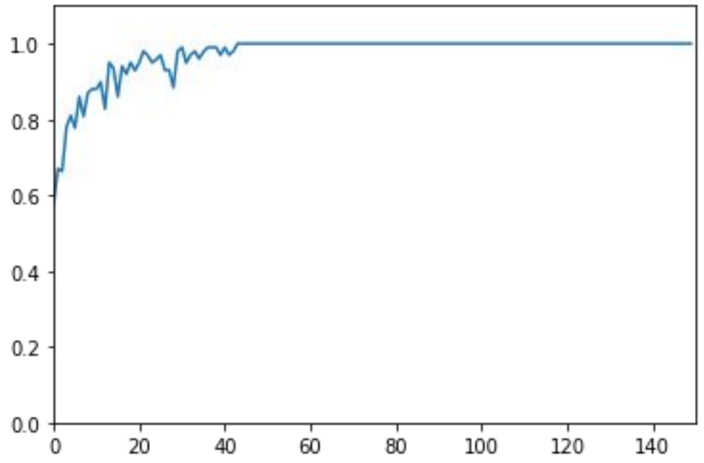


```
global tau1
global tau2
if infected:
    # Infected ant behavior: ignore path and head towards colony
    tau1 += Q / l1
    tau2 += Q / l2
else:
    if path == 1:
        tau1 += Q / l1
    else:
        tau2 += Q / l2
```

Normal Ant Colony Optimization



Fungal Pandemic Ant Optimization



Percentage of the ants using the short path.

As can be seen that **Normal Ant Colony Optimization** often **fails to converge to optimum (reaching 1)**

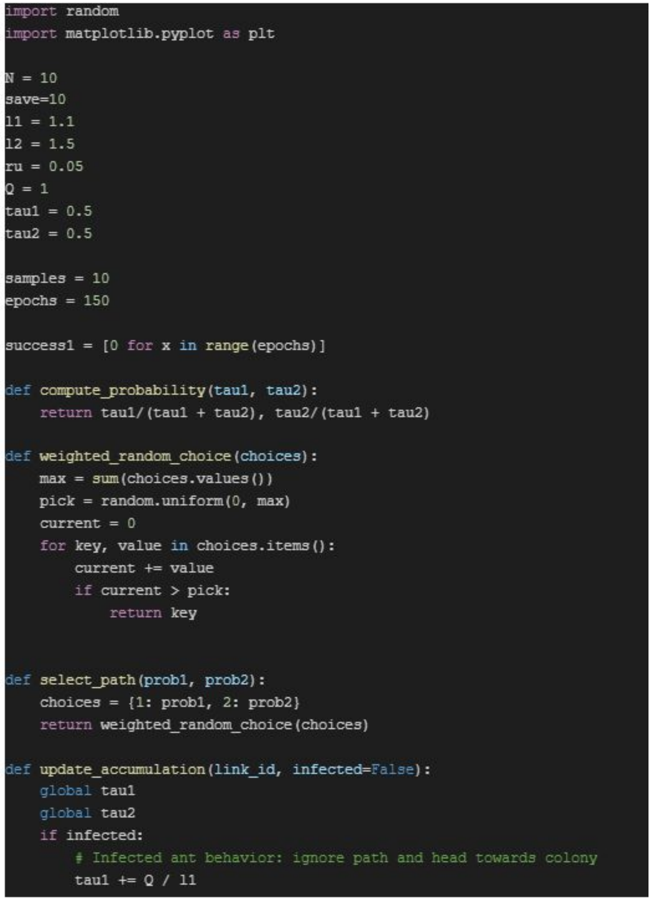
while the **FPAO** converges to the optimum but is a **more computationally expensive and gradual process**

Conclusion

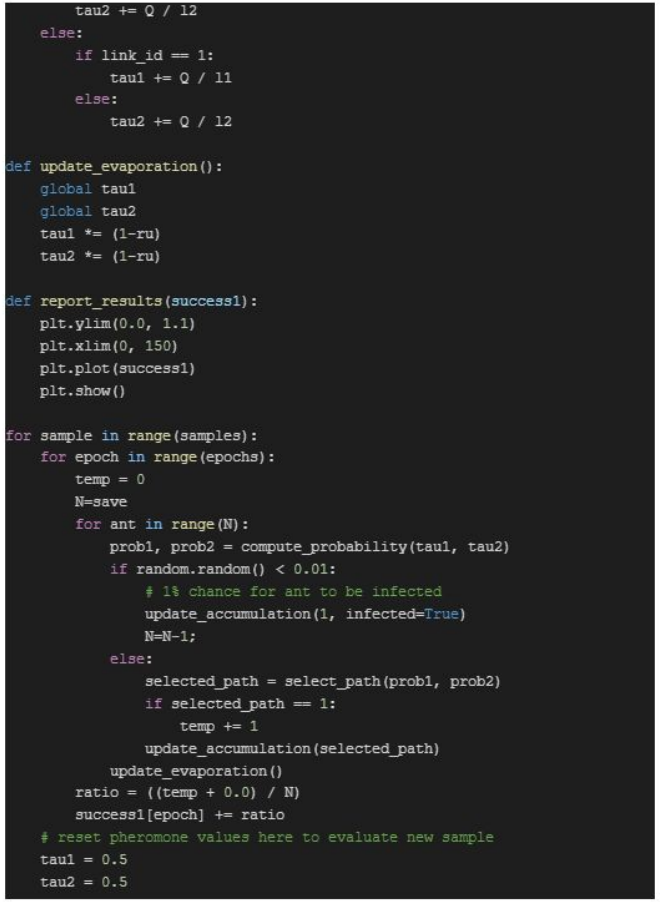
In conclusion, the **Fungal Pandemic Ant Optimization (FPAO)** algorithm is a metaheuristic methods that uses artificial ants to explore the search space and find optimal solutions. **The FPAO algorithm introduces an additional mechanism inspired by the behavior of the cordyceps fungus, which allows infected ants to modify the pheromone trails and help the algorithm escape from local optima.**

The FPAO algorithm may take longer to converge, but it has a higher chance of finding the global optimum.

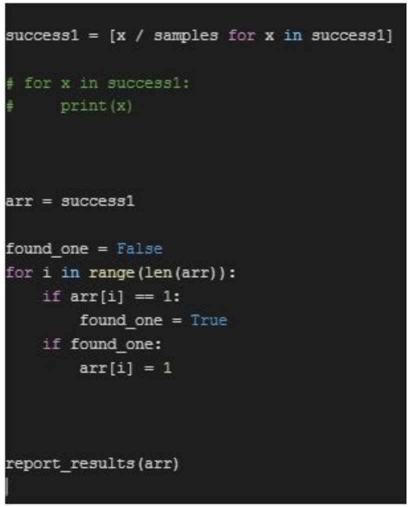
Appendix: Fungal Pandemic Ant Optimization Code



```python
import random
import matplotlib.pyplot as plt

N = 10
save=10
l1 = 1.1
l2 = 1.5
ru = 0.05
Q = 1
tau1 = 0.5
tau2 = 0.5

samples = 10
epochs = 150

success1 = [0 for x in range(epochs)]

def compute_probability(tau1, tau2):
    return tau1/(tau1 + tau2), tau2/(tau1 + tau2)

def weighted_random_choice(choices):
    max = sum(choices.values())
    pick = random.uniform(0, max)
    current = 0
    for key, value in choices.items():
        current += value
        if current > pick:
            return key


def select_path(prob1, prob2):
    choices = {1: prob1, 2: prob2}
    return weighted_random_choice(choices)

def update_accumulation(link_id, infected=False):
    global tau1
    global tau2
    if infected:
        # Infected ant behavior: ignore path and head towards colony
        tau1 += Q / l1
```



```python
        tau2 += Q / l2
    else:
        if link_id == 1:
            tau1 += Q / l1
        else:
            tau2 += Q / l2

def update_evaporation():
    global tau1
    global tau2
    tau1 *= (1-ru)
    tau2 *= (1-ru)

def report_results(success1):
    plt.ylim(0.0, 1.1)
    plt.xlim(0, 150)
    plt.plot(success1)
    plt.show()

for sample in range(samples):
    for epoch in range(epochs):
        temp = 0
        N=save
        for ant in range(N):
            prob1, prob2 = compute_probability(tau1, tau2)
            if random.random() < 0.01:
                # 1% chance for ant to be infected
                update_accumulation(1, infected=True)
                N=N-1;
            else:
                selected_path = select_path(prob1, prob2)
                if selected_path == 1:
                    temp += 1
                update_accumulation(selected_path)
            update_evaporation()
        ratio = ((temp + 0.0) / N)
        success1[epoch] += ratio
    # reset pheromone values here to evaluate new sample
    tau1 = 0.5
    tau2 = 0.5
```



```python
success1 = [x / samples for x in success1]

# for x in success1:
#     print(x)



arr = success1

found_one = False
for i in range(len(arr)):
    if arr[i] == 1:
        found_one = True
    if found_one:
        arr[i] = 1



report_results(arr)
```