Information
Technology
Institute

HONEYBEE MATING OPTIMIZATION (HBMO)

PRESENTED BY
ROLA HOSSAM & MUHAMMAD NASSER
SUPERVISED BY
DR. ENG. GEORGE ISKANDER

PRESENTATION CONTENTS:

INTRODUCTION

- Prof. Dr. Dervis Karaboga and Prof. Dr. Bahriye Basturk, two researchers proposed the Honeybee Mating Optimization (HBMO) algorithm in 2005 in their paper titled:
  "A Powerful and Efficient Algorithm for Numerical Function Optimization: Artificial Bee Colony (ABC) Algorithm"

- The meta-heuristic HBMO algorithm is based on observation of the process of honeybee mating, which involves a combination of local and global search strategies.

INTRODUCTION (contd.)

- In a honeybee colony, drones mate with the queen bee during a process called the “Nuptial Flight". The drones form a swarm and follow the queen bee as she flies and try to mate with the queen bee. After the nuptial flight, the drones die, and the queen bee returns to the colony to lay eggs and produce new workers.

INTRODUCTION (contd.)

- The HBMO algorithm starts with a swarm of male bees (Potential Solutions), which follow the queen bee (Optimal Solution) in the search space.
- The male bees use local and global search strategies to explore the search space and find the optimal solution.
- The algorithm mimics the natural selection process by allowing the best solutions to survive and reproduce, while eliminating the weaker solutions.

META-HEURISTIC

- The HBMO algorithm is a meta-heuristic function which is a high-level problem-solving strategies that are designed to find good solutions to optimization problems.
- Even in cases where the problem is complex, or the solution space is large.
- Meta-heuristics do not guarantee finding the optimal solution, but they are often able to find near-optimal solutions in a reasonable amount of time.

HBMO ALGORITHM

1. **POPULATION INITIALIZATION:**
   - Generate an initial population (Bees). The population is represented as a swarm of male bees, and each bee in the population represents a potential solution.

2. **FITNESS EVALUATION:**
   - Evaluate the fitness of each solution in the population using a fitness function and calculates a score that represents how well the given bee satisfies the objective of the HBMO problem. Such as in the Traveling Salesman Problem (TSP), the fitness function evaluates the total distance or cost of the path taken by the traveling salesman.

3. **QUEEN BEE INITIALIZATION:**
   - Select the bee with the best fitness as the queen bee. It represents the current best solution for the HBMO problem.

HBMO ALGORITHM (contd.)

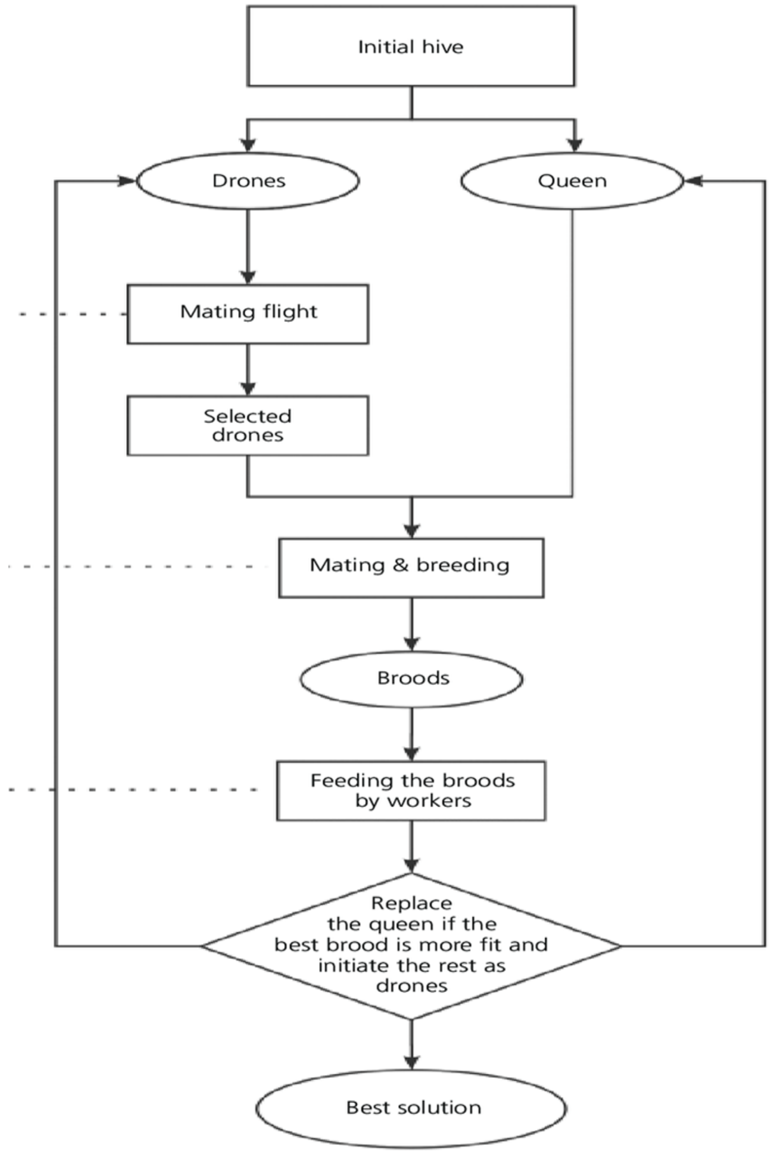
4. **EXPLORATION PHASE:**
   - Each bee performs local search to explore its immediate neighborhood and find the best possible solution in that region.

5. **EXPLOITATION PHASE:**
   - The bees share information about the best solutions they have found so far and use this information to refine their search. Bees start to converge towards the optimal solution.

6. **UPDATE QUEEN BEE:**
   - If a bee finds a solution that is better than the current queen bee, the queen bee is replaced with the new solution.

7. **REPEAT STEPS 4-6 UNTIL A STOPPING CONDITION HAPPENS:**
   - Maximum number of iterations.
   - Satisfactory solution is found.
   - Same solution is found for specific number of times.

HBMO ALGORITHM (contd.)


Initial hive
Drones
Queen
Mating flight
Selected drones
Mating & breeding
Broods
Feeding the broods by workers
Replace the queen if the best brood is more fit and initiate the rest as drones
Best solution

REAL LIFE APPLICATIONS

ENGINEERING

MACHINE LEARNING

FINANCES

IMAGE PROCESSING

HBMO LIMITATIONS

1. **PARAMETER TUNING:**
    - HBMO's performance is sensitive to the choice of its parameters, such as the population size, the number of iterations, and the search strategies. Tuning these parameters can be time-consuming and challenging. It requires domain-specific knowledge.

2. **PREMATURE CONVERGENCE:**
    - HBMO may converge prematurely to a sub-optimal solution if the search space is too narrow, or the algorithm gets trapped in a local optimum. To mitigate this, use a diverse population of male bees (Potential Solutions) and adapting the search strategies to explore the search space effectively.

HBMO LIMITATIONS (contd.)

3. **SCALABILITY:**
   - HBMO may face scalability issues when the problem size is too large, or the search space is too complex. To mitigate this, use a parallel computing techniques or hybridizing HBMO with other optimization algorithms.

4. **STOCHASTICITY:**
   - The stochastic nature of HBMO can lead to instability and unpredictability in its performance, particularly in noisy or uncertain environments. To mitigate this, use a robust optimization techniques or modifying the algorithm to handle uncertainty explicitly.

BENCHMARK FUNCTIONS

- They are mathematical functions that are commonly used to evaluate the performance of optimization algorithms.
- These functions are designed to simulate the types of optimization problems.

- **SPHERE FUNCTION:**
  - The Sphere function is a simple, convex function that has a single global minimum at the origin. It has no local minima or other complicated features.

- **RASTRIGIN FUNCTION:**
  - The Rastrigin function has many local minima and is characterized by many oscillations and its non-convex shape.
  - It is difficult for optimization algorithms to identify the global minimum among the many local minima.

- **ROSENBROCK FUNCTION:**
  - The Rosenbrock function is characterized by a narrow valley that curves towards the minimum, making it difficult for optimization algorithms to traverse.

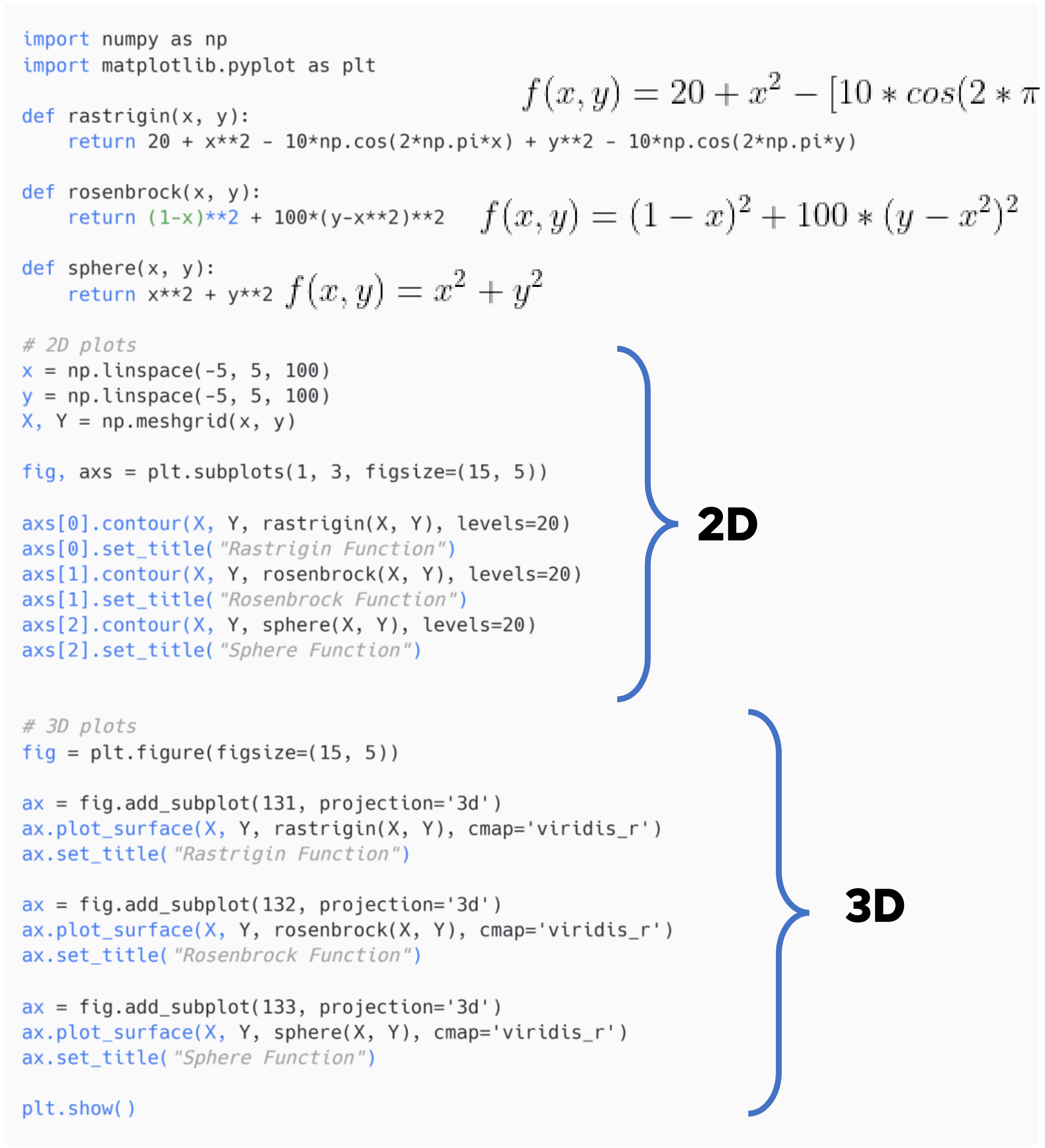
BENCHMARK FUNCTIONS (contd.)

SPHERE Benchmark Function
x = 0.074152 y = 0.01960
Fitness score: 0.00588

ROSENBROCK Benchmark Function
x = 1.02081 y = 1.04065,
Fitness score: 0.00063

RASTRIGIN Benchmark Function
x = 0.041971y = 0.02965,
Fitness score: 0.52143

BENCHMARK FUNCTIONS (contd.)



```python
import numpy as np
import matplotlib.pyplot as plt

def rastrigin(x, y):
    return 20 + x**2 - 10*np.cos(2*np.pi*x) + y**2 - 10*np.cos(2*np.pi*y)

def rosenbrock(x, y):
    return (1-x)**2 + 100*(y-x**2)**2

def sphere(x, y):
    return x**2 + y**2

# 2D plots
x = np.linspace(-5, 5, 100)
y = np.linspace(-5, 5, 100)
X, Y = np.meshgrid(x, y)

fig, axs = plt.subplots(1, 3, figsize=(15, 5))

axs[0].contour(X, Y, rastrigin(X, Y), levels=20)
axs[0].set_title("Rastrigin Function")
axs[1].contour(X, Y, rosenbrock(X, Y), levels=20)
axs[1].set_title("Rosenbrock Function")
axs[2].contour(X, Y, sphere(X, Y), levels=20)
axs[2].set_title("Sphere Function")

# 3D plots
fig = plt.figure(figsize=(15, 5))

ax = fig.add_subplot(131, projection='3d')
ax.plot_surface(X, Y, rastrigin(X, Y), cmap='viridis_r')
ax.set_title("Rastrigin Function")

ax = fig.add_subplot(132, projection='3d')
ax.plot_surface(X, Y, rosenbrock(X, Y), cmap='viridis_r')
ax.set_title("Rosenbrock Function")

ax = fig.add_subplot(133, projection='3d')
ax.plot_surface(X, Y, sphere(X, Y), cmap='viridis_r')
ax.set_title("Sphere Function")

plt.show()
```

$$f(x,y) = 20 + x^2 - [10 * cos(2 * \pi * x))] + y^2 - [10 * cos(2 * \pi * y)]$$

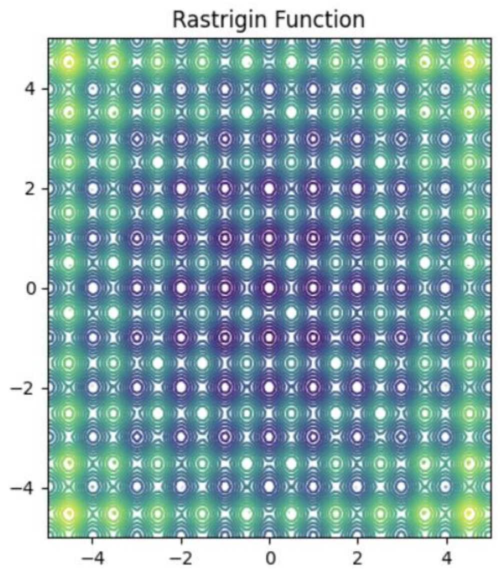
$$f(x,y) = (1-x)^2 + 100 * (y - x^2)^2$$

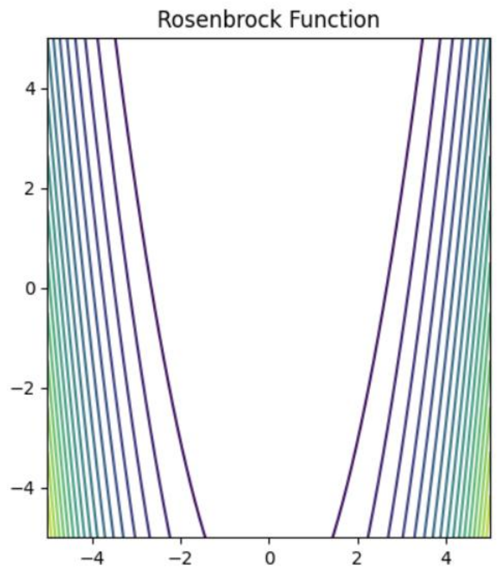
$$f(x,y) = x^2 + y^2$$

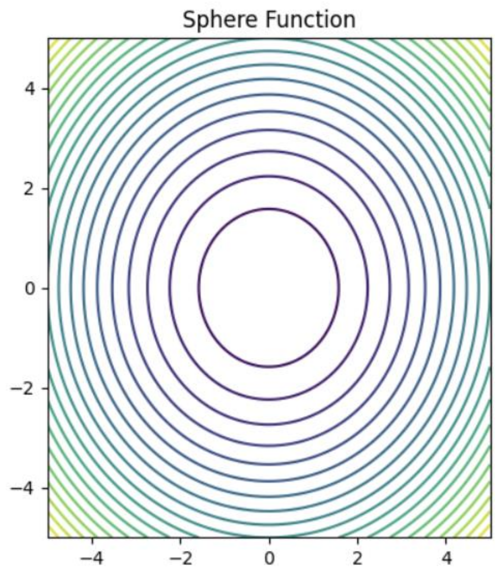
2D

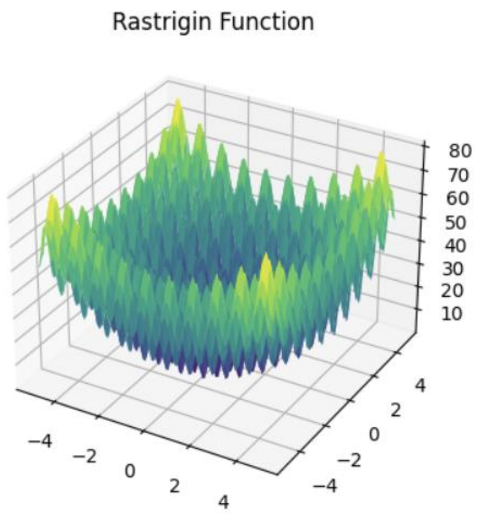
3D

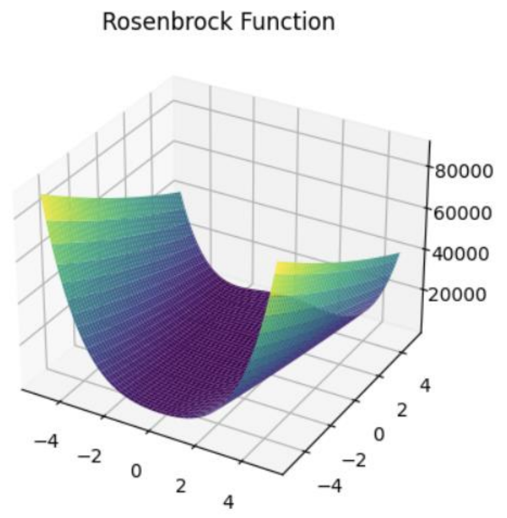
BENCHMARK FUNCTIONS (contd.)

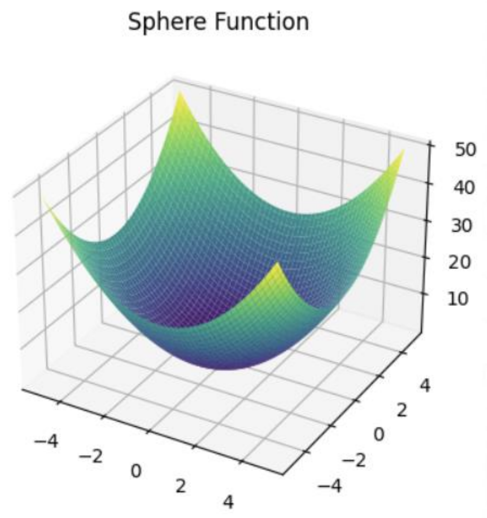




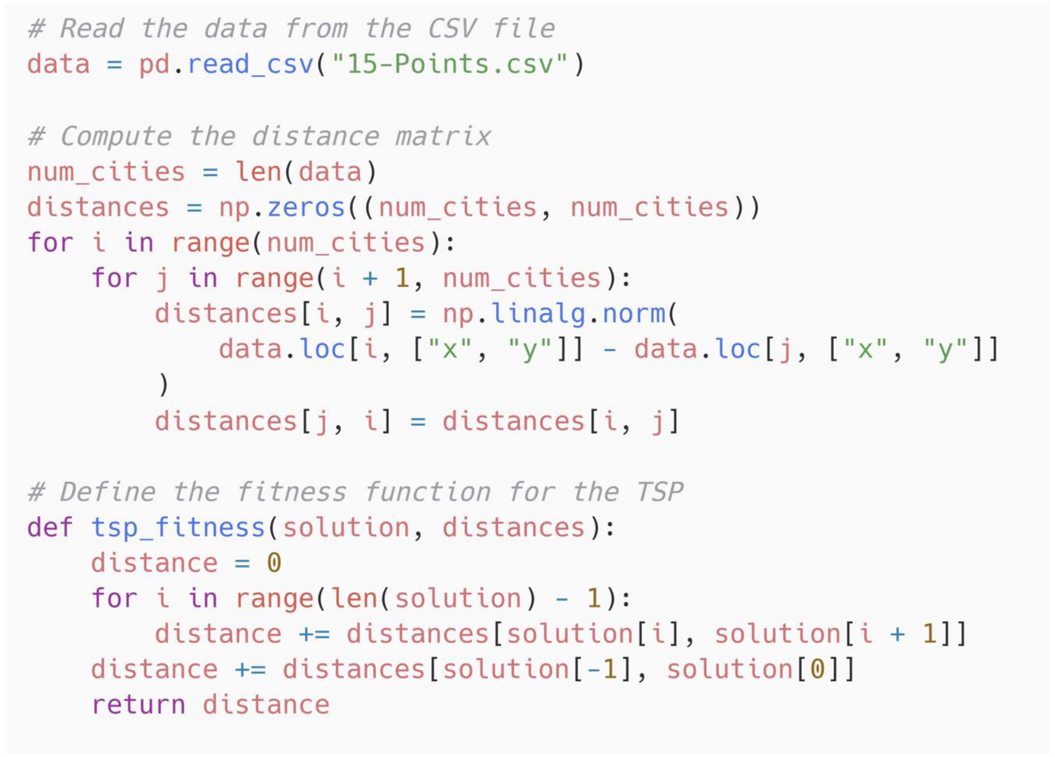








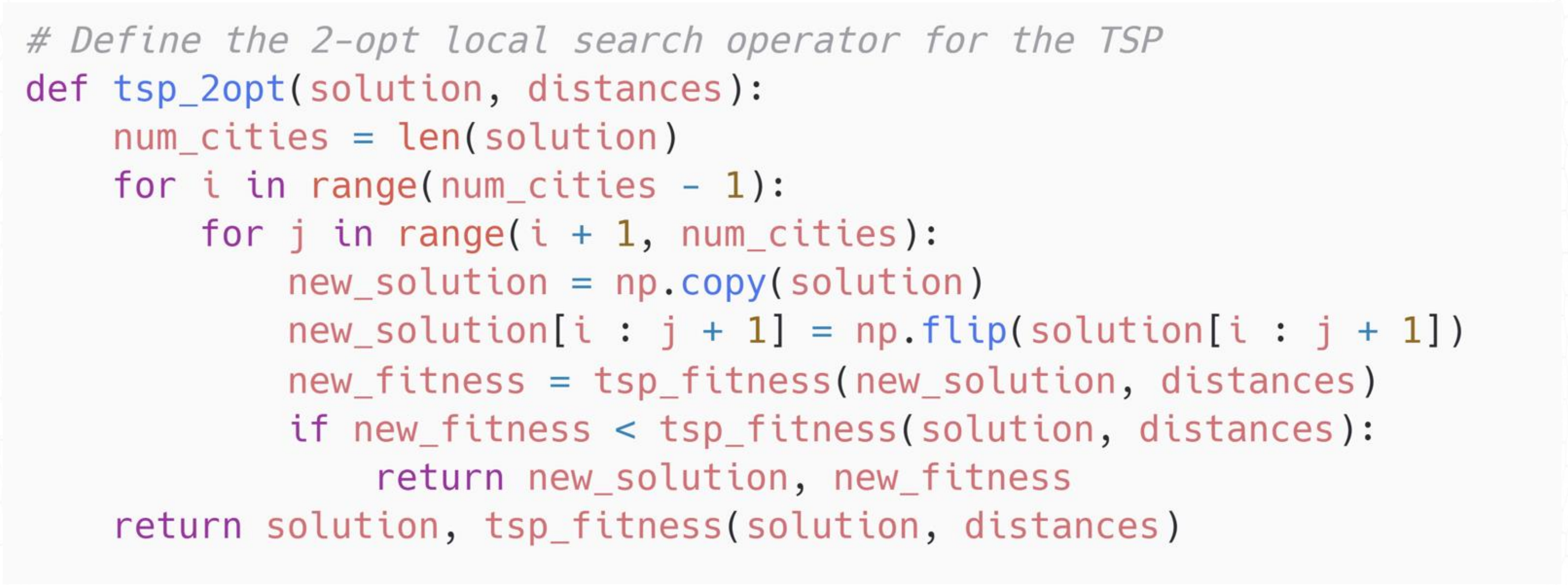
HBMO IMPLEMENTATION



```python
# Read the data from the CSV file
data = pd.read_csv("15-Points.csv")

# Compute the distance matrix
num_cities = len(data)
distances = np.zeros((num_cities, num_cities))
for i in range(num_cities):
    for j in range(i + 1, num_cities):
        distances[i, j] = np.linalg.norm(
            data.loc[i, ["x", "y"]] - data.loc[j, ["x", "y"]]
        )
        distances[j, i] = distances[i, j]

# Define the fitness function for the TSP
def tsp_fitness(solution, distances):
    distance = 0
    for i in range(len(solution) - 1):
        distance += distances[solution[i], solution[i + 1]]
    distance += distances[solution[-1], solution[0]]
    return distance
```

This part of code computes the distance function using Euclidean distance for 15 cities and using it in a defined function to compute the different solutions of fitness function.

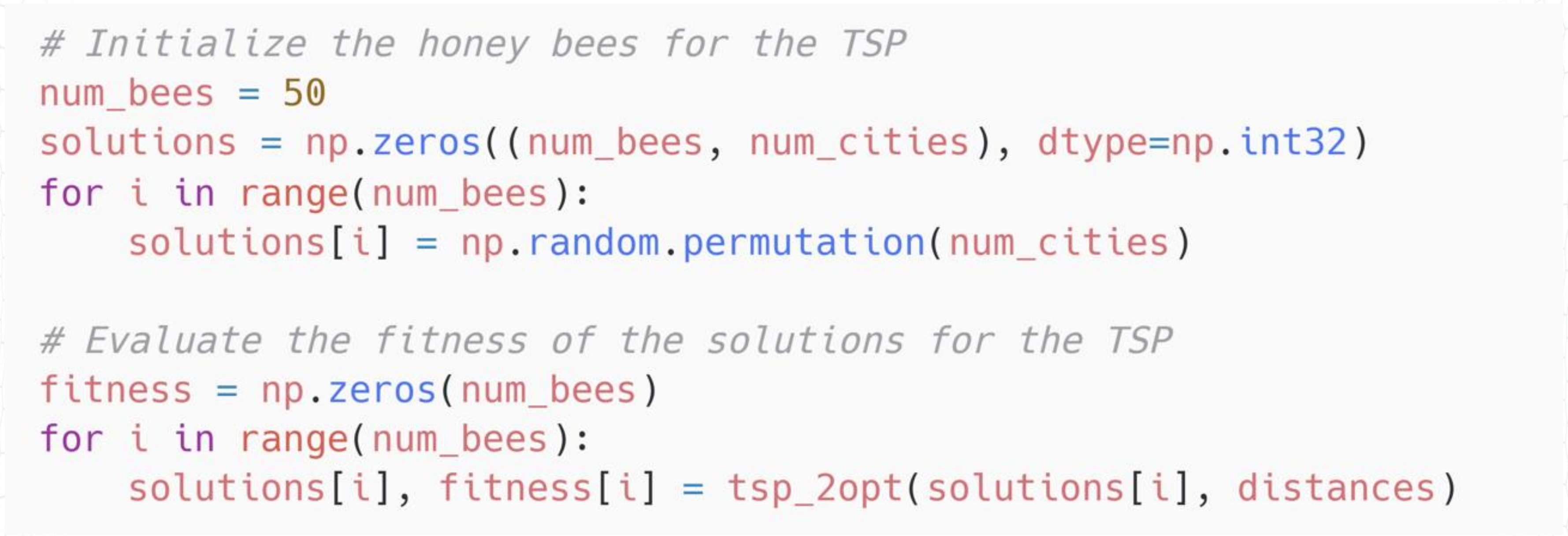
HBMO IMPLEMENTATION (contd.)



```python
# Define the 2-opt local search operator for the TSP
def tsp_2opt(solution, distances):
    num_cities = len(solution)
    for i in range(num_cities - 1):
        for j in range(i + 1, num_cities):
            new_solution = np.copy(solution)
            new_solution[i : j + 1] = np.flip(solution[i : j + 1])
            new_fitness = tsp_fitness(new_solution, distances)
            if new_fitness < tsp_fitness(solution, distances):
                return new_solution, new_fitness
    return solution, tsp_fitness(solution, distances)
```

This part of code takes the two best local optima and generates new solutions using the same distance function and to reduce the total distance it flips the edges.

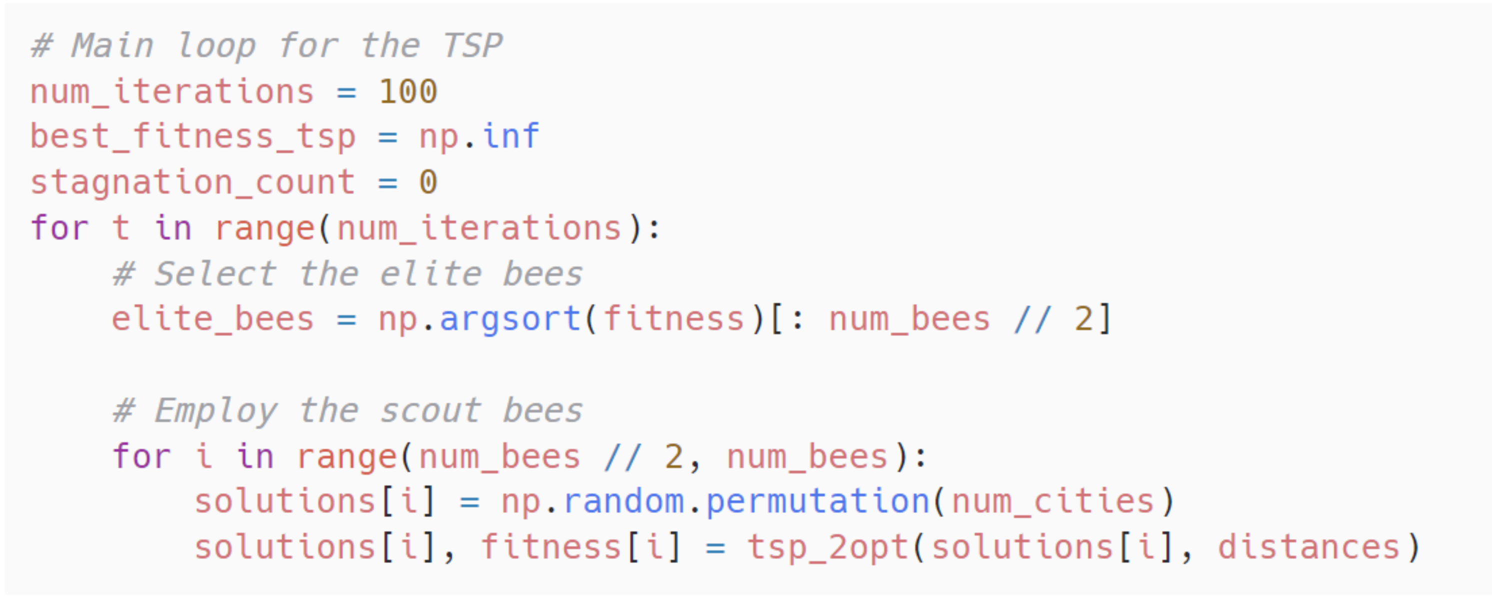
HBMO IMPLEMENTATION (contd.)



```python
# Initialize the honey bees for the TSP
num_bees = 50
solutions = np.zeros((num_bees, num_cities), dtype=np.int32)
for i in range(num_bees):
    solutions[i] = np.random.permutation(num_cities)

# Evaluate the fitness of the solutions for the TSP
fitness = np.zeros(num_bees)
for i in range(num_bees):
    solutions[i], fitness[i] = tsp_2opt(solutions[i], distances)
```

This part of code illustrates the known steps of swarm intelligence algorithms we initialized the honey bees as 50 bees. And the second step is evaluating the fitness function of the solutions for the TSP.

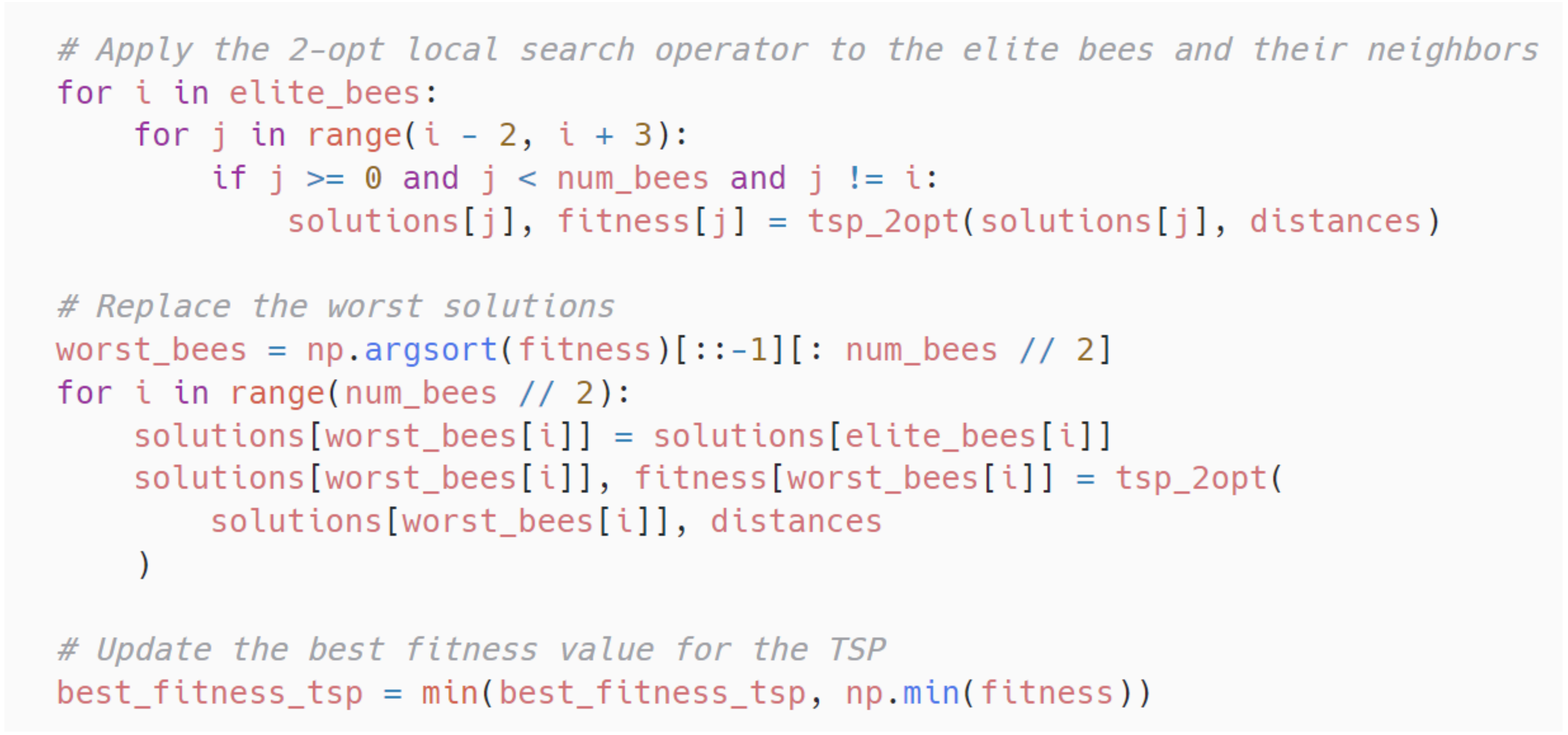
HBMO IMPLEMENTATION (contd.)



```python
# Main loop for the TSP
num_iterations = 100
best_fitness_tsp = np.inf
stagnation_count = 0
for t in range(num_iterations):
    # Select the elite bees
    elite_bees = np.argsort(fitness)[: num_bees // 2]

    # Employ the scout bees
    for i in range(num_bees // 2, num_bees):
        solutions[i] = np.random.permutation(num_cities)
        solutions[i], fitness[i] = tsp_2opt(solutions[i], distances)
```

This part of code runs for 100 iterations selecting the elite - scouts bees. The elite bees exploit promising solutions in the search space, while the scout bees explore new areas of the search space to avoid getting stuck in local minima.

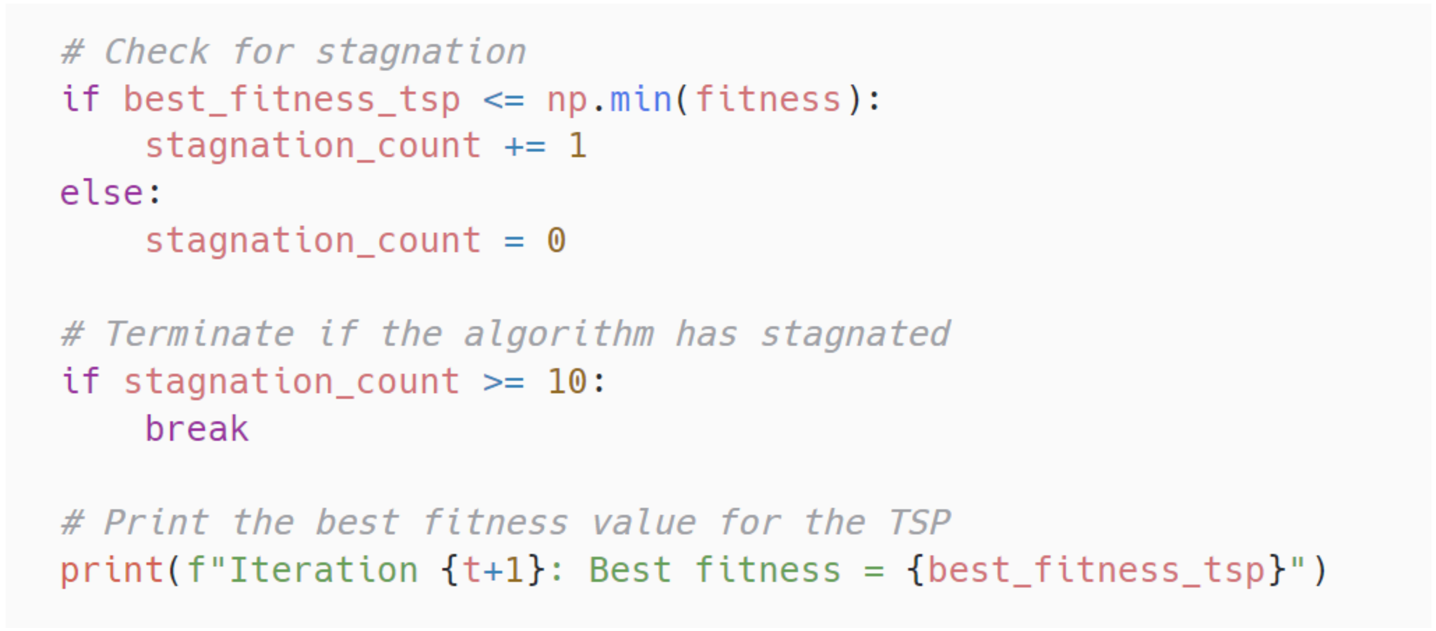
HBMO IMPLEMENTATION (contd.)



```python
# Apply the 2-opt local search operator to the elite bees and their neighbors
for i in elite_bees:
    for j in range(i - 2, i + 3):
        if j >= 0 and j < num_bees and j != i:
            solutions[j], fitness[j] = tsp_2opt(solutions[j], distances)

# Replace the worst solutions
worst_bees = np.argsort(fitness)[::-1][: num_bees // 2]
for i in range(num_bees // 2):
    solutions[worst_bees[i]] = solutions[elite_bees[i]]
    solutions[worst_bees[i]], fitness[worst_bees[i]] = tsp_2opt(
        solutions[worst_bees[i]], distances
    )

# Update the best fitness value for the TSP
best_fitness_tsp = min(best_fitness_tsp, np.min(fitness))
```

This part of code chooses the best 2 local optima and replacing the worst bees with the best ones and the last step is to update the best fitness value (minimum value) for the TSP .

HBMO IMPLEMENTATION (contd.)

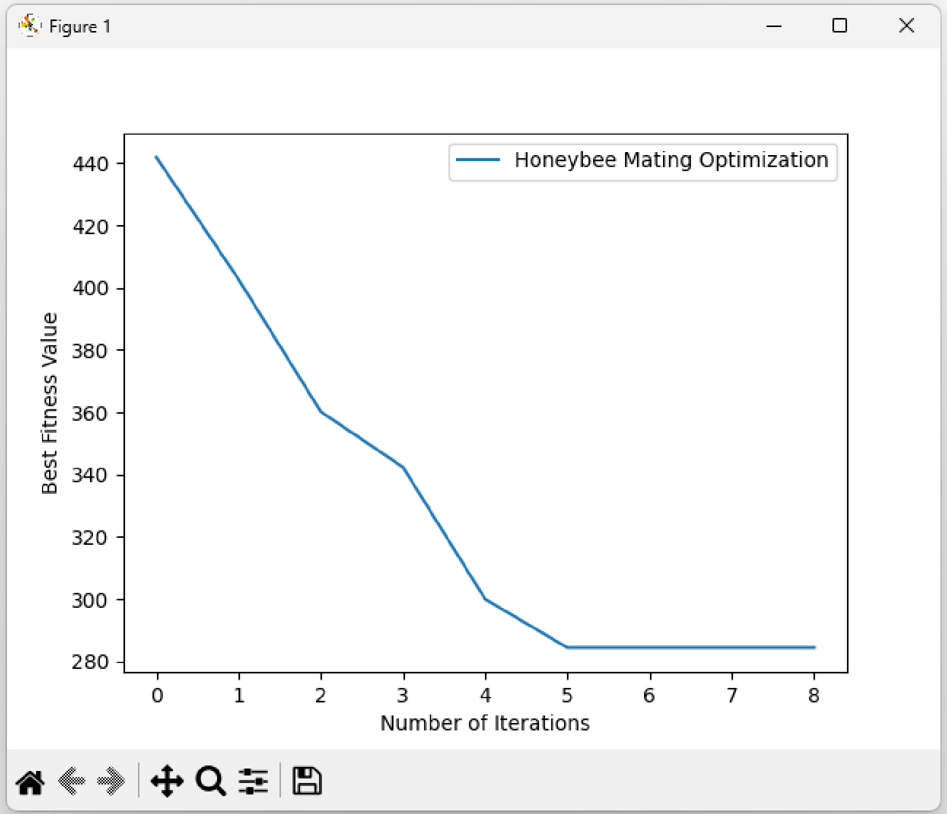


```python
# Check for stagnation
if best_fitness_tsp <= np.min(fitness):
    stagnation_count += 1
else:
    stagnation_count = 0

# Terminate if the algorithm has stagnated
if stagnation_count >= 10:
    break

# Print the best fitness value for the TSP
print(f"Iteration {t+1}: Best fitness = {best_fitness_tsp}")
```

This part of code checks if the algorithm getting stuck in the same point (or can't update any more) and stop the loop and then print the best fitness tsp.

HBMO-TSP OPTIMAL SOLUTION

OPTIMAL SOLUTION REACHED & CONFIRMED AFTER 9 ITERATIONS...



HBMO-TSP OPTIMAL SOLUTION

TSP (DISTANCE) FITNESS: 284.3810904080331

REFERENCES

- Karaboga, D., Basturk, B. A powerful and efficient algorithm for numerical function optimization: artificial bee colony (ABC) algorithm. J Glob Optim 39, 459–471 (2007). https://doi.org/10.1007/s10898-007-9149-x
- https://beekeepclub.com/what-is-the-honeybee-queen-mating-flight/
- https://www.researchgate.net/publication/259603599_Honey-Bees_Mating_Optimization_HBMO_Algorithm_A_New_Heuristic_Approach_for_Water_Resources_Optimization
- https://ascelibrary.org/doi/10.1061/40976%28316%29496
- https://en.wikipedia.org/wiki/Test_functions_for_optimization

Information Technology Institute

THANK YOU

FOR YOUR ATTENTION

ROLA HOSSAM & MUHAMMAD NASSER