Program 8

Problem statement:Vehicle Routing Using Ant Colony Optimization

Code:

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

num_vehicles = 5
num_iterations = 100
alpha = 1.0
beta = 2.0
evaporation_rate = 0.5
graph = [
    [0, 2, 4, 1],
    [2, 0, 1, 5],
    [4, 1, 0, 3],
    [1, 5, 3, 0]
]
num_cities = len(graph)
pheromone = [[1.0 for _ in range(num_cities)] for _ in range(num_cities)]
best_path = None
best_distance = float('inf')

pheromone_history = []
paths_history = []

for iteration in range(num_iterations):
    paths = []
    for _ in range(num_vehicles):
        start_city = random.randint(0, num_cities - 1)
        path = [start_city]
        visited = [False] * num_cities
        visited[start_city] = True
        for _ in range(num_cities - 1):
            current_city = path[-1]
            next_city = None
            probabilities = []
            for city in range(num_cities):
                if not visited[city]:
                    pheromone_value = pheromone[current_city][city]
                    heuristic_value = 1.0 / graph[current_city][city]
                    probability = pheromone_value * alpha * heuristic_value * beta
                    probabilities.append((city, probability))

            total_probability = sum(prob for _, prob in probabilities)
            probabilities = [(city, prob / total_probability) for city, prob in probabilities]
```

```
            random_value = random.uniform(0, 1)
            cumulative_probability = 0.0
            for city, probability in probabilities:
                cumulative_probability += probability
                if random_value <= cumulative_probability:
                    next_city = city
                    break
            path.append(next_city)
            visited[next_city] = True

        paths.append(path)

    for i in range(num_cities):
        for j in range(num_cities):
            if i != j:
                pheromone[i][j] *= (1 - evaporation_rate)

    for path in paths:
        distance = sum(graph[path[i]][path[i+1]] for i in range(num_cities - 1))
        if distance < best_distance:
            best_distance = distance
            best_path = path

        for i in range(num_cities - 1):
            pheromone[path[i]][path[i+1]] += 1.0 / distance

    pheromone_history.append(np.array(pheromone))
    paths_history.append(paths)

fig, ax = plt.subplots(figsize=(8, 6))

best_x = [p % 2 for p in best_path]
best_y = [p // 2 for p in best_path]
ax.plot(best_x, best_y, marker='o', color='red', linestyle='-', linewidth=2, label="Best Path")
for path in paths_history[-1]:
    x = [p % 2 for p in path]
    y = [p // 2 for p in path]
    ax.plot(x, y, marker='o', label="Ant Path", alpha=0.7)

ax.set_title(f"Ant Paths at Iteration {num_iterations}")
ax.set_xlabel("X")
ax.set_ylabel("Y")
plt.legend()
plt.show()

print("Best Path:", best_path)
```

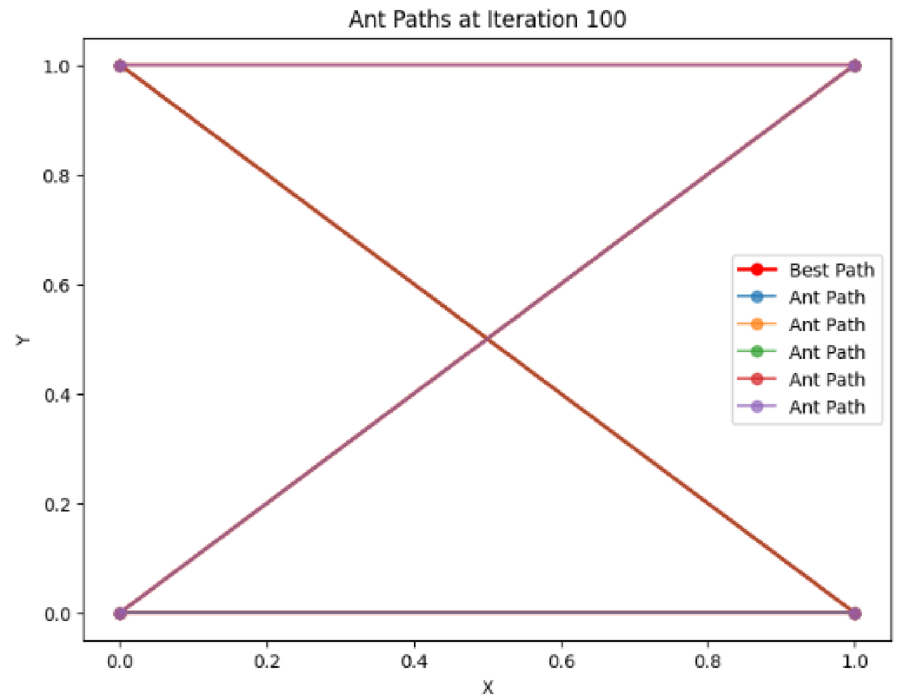
print("Distance:", best_distance)

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

num_vehicles = 5
num_iterations = 100
alpha = 1.0
beta = 2.0
evaporation_rate = 0.5
graph = [
    [0, 2, 4, 1],
    [2, 0, 1, 5],
    [4, 1, 0, 3],
    [1, 5, 3, 0]
]
num_cities = len(graph)
pheromone = [[1.0 for _ in range(num_cities)] for _ in range(num_cities)]
best_path = None
best_distance = float('inf')

pheromone_history = []
paths_history = []

for iteration in range(num_iterations):
    paths = []
    for _ in range(num_vehicles):
        start_city = random.randint(0, num_cities - 1)
        path = [start_city]
        visited = [False] * num_cities
        visited[start_city] = True
        for _ in range(num_cities - 1):
            current_city = path[-1]
            next_city = None
            probabilities = []
            for city in range(num_cities):
                if not visited[city]:
                    pheromone_value = pheromone[current_city][city]
```

```python
                heuristic_value = 1.0 / graph[current_city][city]
                probability = pheromone_value * alpha * heuristic_value * beta
                probabilities.append((city, probability))

            total_probability = sum(prob for _, prob in probabilities)
            probabilities = [(city, prob / total_probability) for city, prob in probabilities]
            random_value = random.uniform(0, 1)
            cumulative_probability = 0.0
            for city, probability in probabilities:
                cumulative_probability += probability
                if random_value <= cumulative_probability:
                    next_city = city
                    break
            path.append(next_city)
            visited[next_city] = True

        paths.append(path)

    for i in range(num_cities):
        for j in range(num_cities):
            if i != j:
                pheromone[i][j] *= (1 - evaporation_rate)

    for path in paths:
        distance = sum(graph[path[i]][path[i+1]] for i in range(num_cities - 1))
        if distance < best_distance:
            best_distance = distance
            best_path = path

        for i in range(num_cities - 1):
            pheromone[path[i]][path[i+1]] += 1.0 / distance

    pheromone_history.append(np.array(pheromone))
    paths_history.append(paths)

fig, ax = plt.subplots(figsize=(8, 6))
```

```python
best_x = [p % 2 for p in best_path]
best_y = [p // 2 for p in best_path]
ax.plot(best_x, best_y, marker='o', color='red', linestyle='-', linewidth=2, label="Best Path")
for path in paths_history[-1]:
    x = [p % 2 for p in path]
    y = [p // 2 for p in path]
    ax.plot(x, y, marker='o', label="Ant Path", alpha=0.7)

ax.set_title(f"Ant Paths at Iteration {num_iterations}")
ax.set_xlabel("X")
ax.set_ylabel("Y")
plt.legend()
plt.show()

print("Best Path:", best_path)
print("Distance:", best_distance)
```

Output:




```
Best Path: [3, 0, 1, 2]
Distance: 4
```