Ant Colony Optimization (ACO)

```python
import numpy as np

# Distance function between cities
def distance(city1, city2):
    return np.linalg.norm(np.array(city1) - np.array(city2))

# Total length of a tour
def tour_length(tour, cities):
    dist = 0
    for i in range(len(tour)):
        dist += distance(cities[tour[i]], cities[tour[(i + 1) % len(tour)]])
    return dist

# Ant Colony Optimization for TSP
def ant_colony_optimization(cities, n_ants=10, n_iter=100, alpha=1, beta=5, rho=0.5, initial_pheromone=1
    .0):
    n_cities = len(cities)
    # Initialize pheromone matrix
    pheromone = np.ones((n_cities, n_cities)) * initial_pheromone

    # Precompute heuristic info (1/distance)
    heuristic = np.zeros((n_cities, n_cities))
    for i in range(n_cities):
        for j in range(n_cities):
            if i != j:
                heuristic[i, j] = 1.0 / distance(cities[i], cities[j])

    best_tour = None
    best_length = float("inf")
```

Ant Colony Optimization (ACO)

```
for iteration in range(n_iter):
    all_tours = []
    all_lengths = []

    # Each ant constructs a tour
    for ant in range(n_ants):
        visited = [np.random.randint(0, n_cities)]
        while len(visited) < n_cities:
            current = visited[-1]
            probs = []
            for next_city in range(n_cities):
                if next_city not in visited:
                    tau = pheromone[current, next_city] ** alpha
                    eta = heuristic[current, next_city] ** beta
                    probs.append(tau * eta)
                else:
                    probs.append(0)
            probs = np.array(probs)
            probs /= probs.sum()
            next_city = np.random.choice(range(n_cities), p=probs)
            visited.append(next_city)
```

Ant Colony Optimization (ACO)

```
            all_tours.append(visited)
            length = tour_length(visited, cities)
            all_lengths.append(length)

            if length < best_length:
                best_length = length
                best_tour = visited.copy()

        # Pheromone evaporation
        pheromone *= (1 - rho)

        # Pheromone update
        for tour, length in zip(all_tours, all_lengths):
            for i in range(n_cities):
                from_city = tour[i]
                to_city = tour[(i + 1) % n_cities]
                pheromone[from_city, to_city] += 1.0 / length

        print(f"Iteration {iteration+1}: Best Length = {best_length:.4f}")

    return best_tour, best_length

# Example usage with coordinates of cities
cities = [(0,0), (1,5), (5,2), (6,6), (8,3)]
best_tour, best_length = ant_colony_optimization(cities, n_ants=20, n_iter=50)

print("\nBest Tour:", best_tour)
print("Best Tour Length:", best_length)
```

Ant Colony Optimization (ACO)

Output

```
Iteration 1: Best Length = 22.3510
Iteration 2: Best Length = 22.3510
Iteration 3: Best Length = 22.3510
Iteration 4: Best Length = 22.3510
Iteration 5: Best Length = 22.3510
Iteration 6: Best Length = 22.3510
Iteration 7: Best Length = 22.3510
Iteration 8: Best Length = 22.3510
Iteration 9: Best Length = 22.3510
Iteration 10: Best Length = 22.3510
Iteration 11: Best Length = 22.3510
Iteration 12: Best Length = 22.3510
Iteration 13: Best Length = 22.3510
Iteration 14: Best Length = 22.3510
Iteration 15: Best Length = 22.3510
Iteration 16: Best Length = 22.3510
Iteration 17: Best Length = 22.3510
Iteration 18: Best Length = 22.3510
Iteration 19: Best Length = 22.3510
Iteration 20: Best Length = 22.3510
Iteration 21: Best Length = 22.3510
Iteration 22: Best Length = 22.3510
Iteration 23: Best Length = 22.3510
Iteration 24: Best Length = 22.3510
Iteration 25: Best Length = 22.3510
Iteration 26: Best Length = 22.3510
Iteration 27: Best Length = 22.3510
Iteration 28: Best Length = 22.3510
Iteration 29: Best Length = 22.3510
Iteration 30: Best Length = 22.3510
Iteration 31: Best Length = 22.3510
Iteration 32: Best Length = 22.3510
Iteration 33: Best Length = 22.3510
Iteration 34: Best Length = 22.3510
Iteration 35: Best Length = 22.3510
Iteration 36: Best Length = 22.3510
```

Ant Colony Optimization (ACO)

```
Best Tour: [0, np.int64(2), np.int64(4), np.int64(3), np.int64(1)]
Best Tour Length: 22.35103276995244
```