```python
import numpy as np
import random

class AntColony:
    def __init__(self, distances, n_ants, n_iterations, alpha=1, beta=2, evaporation_rate=0.5, pheromone_deposit=1.0):
        self.distances = distances
        self.n_ants = n_ants
        self.n_iterations = n_iterations
        self.alpha = alpha  # Influence of pheromone
        self.beta = beta    # Influence of heuristic (1/distance)
        self.evaporation_rate = evaporation_rate
        self.pheromone_deposit = pheromone_deposit
        self.n_cities = distances.shape[0]
        self.pheromone = np.ones((self.n_cities, self.n_cities))  # Initialize pheromone levels

    def run(self):
        best_path = None
        best_path_length = float('inf')

        for iteration in range(self.n_iterations):
            paths, path_lengths = self.construct_solutions()
            self.update_pheromones(paths, path_lengths)

            # Find the best path in this iteration
            min_length = min(path_lengths)
            min_index = path_lengths.index(min_length)

            if min_length < best_path_length:
                best_path_length = min_length
                best_path = paths[min_index]

            print(f"Iteration {iteration + 1}: Best path length = {best_path_length}")

        return best_path, best_path_length

    def construct_solutions(self):
        paths = []
        path_lengths = []

        for _ in range(self.n_ants):
            path = [random.randint(0, self.n_cities - 1)]
            while len(path) < self.n_cities:
                current_city = path[-1]
                next_city = self.select_next_city(current_city, path)
                path.append(next_city)

            # Complete the tour
            path.append(path[0])
            paths.append(path)
            path_lengths.append(self.calculate_path_length(path))

        return paths, path_lengths

    def select_next_city(self, current_city, visited):
        probabilities = []
        for next_city in range(self.n_cities):
            if next_city in visited:
                probabilities.append(0)
            else:
                pheromone = self.pheromone[current_city, next_city] ** self.alpha
                heuristic = (1 / self.distances[current_city, next_city]) ** self.beta
                probabilities.append(pheromone * heuristic)

        probabilities = np.array(probabilities)
        probabilities /= probabilities.sum()

        return np.random.choice(range(self.n_cities), p=probabilities)

    def calculate_path_length(self, path):
        length = 0
        for i in range(len(path) - 1):
            length += self.distances[path[i], path[i + 1]]
        return length

    def update_pheromones(self, paths, path_lengths):
        # Evaporate pheromones
        self.pheromone *= (1 - self.evaporation_rate)

        # Deposit pheromones
        for path, length in zip(paths, path_lengths):
            for i in range(len(path) - 1):
```

```
                self.pheromone[path[i], path[i + 1]] += self.pheromone_deposit / length
                self.pheromone[path[i + 1], path[i]] += self.pheromone_deposit / length
```

```
# Example Usage
if __name__ == "__main__":
    # Example distance matrix (symmetric TSP)
    distances = np.array([
        [0, 2, 2, 5],
        [2, 0, 3, 4],
        [2, 3, 0, 1],
        [5, 4, 1, 0]
    ])

    n_ants = 10
    n_iterations = 50
    alpha = 1
    beta = 2
    evaporation_rate = 0.5
    pheromone_deposit = 1.0

    aco = AntColony(distances, n_ants, n_iterations, alpha, beta, evaporation_rate, pheromone_deposit)
    best_path, best_path_length = aco.run()

    print("\nBest path:", best_path)
    print("Best path length:", best_path_length)
```

```
Iteration 1: Best path length = 9
Iteration 2: Best path length = 9
Iteration 3: Best path length = 9
Iteration 4: Best path length = 9
Iteration 5: Best path length = 9
Iteration 6: Best path length = 9
Iteration 7: Best path length = 9
Iteration 8: Best path length = 9
Iteration 9: Best path length = 9
Iteration 10: Best path length = 9
Iteration 11: Best path length = 9
Iteration 12: Best path length = 9
Iteration 13: Best path length = 9
Iteration 14: Best path length = 9
Iteration 15: Best path length = 9
Iteration 16: Best path length = 9
Iteration 17: Best path length = 9
Iteration 18: Best path length = 9
Iteration 19: Best path length = 9
Iteration 20: Best path length = 9
Iteration 21: Best path length = 9
Iteration 22: Best path length = 9
Iteration 23: Best path length = 9
Iteration 24: Best path length = 9
Iteration 25: Best path length = 9
Iteration 26: Best path length = 9
Iteration 27: Best path length = 9
Iteration 28: Best path length = 9
Iteration 29: Best path length = 9
Iteration 30: Best path length = 9
Iteration 31: Best path length = 9
Iteration 32: Best path length = 9
Iteration 33: Best path length = 9
Iteration 34: Best path length = 9
Iteration 35: Best path length = 9
Iteration 36: Best path length = 9
Iteration 37: Best path length = 9
Iteration 38: Best path length = 9
Iteration 39: Best path length = 9
Iteration 40: Best path length = 9
Iteration 41: Best path length = 9
Iteration 42: Best path length = 9
Iteration 43: Best path length = 9
Iteration 44: Best path length = 9
Iteration 45: Best path length = 9
Iteration 46: Best path length = 9
Iteration 47: Best path length = 9
Iteration 48: Best path length = 9
Iteration 49: Best path length = 9
Iteration 50: Best path length = 9

Best path: [0, 1, 3, 2, 0]
Best path length: 9
```