```
import random

class Particle:
    def __init__(self, dimensions, bounds):
        self.position = [random.uniform(bounds[i][0], bounds[i][1]) for
i in range(dimensions)]
        self.velocity = [random.uniform(-1, 1) for _ in
range(dimensions)]
        self.best_position = self.position[:]
        self.best_fitness = float('inf')


def calculate_fitness(solution, distance_matrix):
    fitness = 0
    for i in range(len(solution) - 1):
      fitness += distance_matrix[int(solution[i])][int(solution[i+1])]
    fitness += distance_matrix[int(solution[-1])][int(solution[0])]
    return fitness


def particle_swarm_optimization(distance_matrix, num_particles,
max_iterations, w, c1, c2):
    num_cities = len(distance_matrix)
    bounds = [(0, num_cities -1)] * num_cities # Each dimension
represents a city index
    particles = [Particle(num_cities, bounds) for _ in
range(num_particles)]
    global_best_position = None
    global_best_fitness = float('inf')

    for iteration in range(max_iterations):
        for particle in particles:
            #Round positions to nearest integer (city index)
            rounded_position = [round(x) for x in particle.position]
            #Encode as a permutation
            solution = [str(i) for i in rounded_position]

            fitness = calculate_fitness(solution, distance_matrix)
            if fitness < particle.best_fitness:
                particle.best_fitness = fitness
                particle.best_position = particle.position[:]

            if fitness < global_best_fitness:
                global_best_fitness = fitness
                global_best_position = particle.position[:]

            for i in range(num_cities):
                r1 = random.random()
                r2 = random.random()
```

```
                particle.velocity[i] = (w * particle.velocity[i] +
                                        c1 * r1 *
(particle.best_position[i] - particle.position[i]) +
                                        c2 * r2 *
(global_best_position[i] - particle.position[i]))

                particle.position[i] += particle.velocity[i]

                #Keep positions within bounds
                particle.position[i] = max(bounds[i][0],
min(bounds[i][1], particle.position[i]))

        print(f"Iteration {iteration+1}: Best Fitness =
{global_best_fitness}")

    rounded_solution = [round(x) for x in global_best_position]
    final_solution = [str(i) for i in rounded_solution]
    print(f"Final solution:{final_solution} - Final
fitness:{global_best_fitness}")


# Example usage
distance_matrix = [
        [0, 2, float('inf'), 12, 5],
        [2, 0, 4, 8, float('inf')],
        [float('inf'), 4, 0, 3, 3],
        [12, 8, 3, 0, 10],
        [5, float('inf'), 3, 10, 0],
    ]

num_particles = 50
max_iterations = 100
w = 0.7 # Inertia weight
c1 = 1.4  # Cognitive component
c2 = 1.4  # Social component

particle_swarm_optimization(distance_matrix, num_particles,
max_iterations, w, c1, c2)
```

Output:

```
Iteration 1: Best Fitness = 6
Iteration 2: Best Fitness = 6
Iteration 3: Best Fitness = 6
Iteration 4: Best Fitness = 0
Iteration 5: Best Fitness = 0
Iteration 6: Best Fitness = 0
Iteration 7: Best Fitness = 0
Iteration 8: Best Fitness = 0
Iteration 9: Best Fitness = 0
Iteration 10: Best Fitness = 0
Iteration 11: Best Fitness = 0
Iteration 12: Best Fitness = 0
Iteration 13: Best Fitness = 0
Iteration 14: Best Fitness = 0
Iteration 15: Best Fitness = 0
Iteration 16: Best Fitness = 0
Iteration 17: Best Fitness = 0
Iteration 18: Best Fitness = 0
Iteration 19: Best Fitness = 0
Iteration 20: Best Fitness = 0
Iteration 21: Best Fitness = 0
Iteration 22: Best Fitness = 0
Iteration 23: Best Fitness = 0
Iteration 24: Best Fitness = 0
Iteration 25: Best Fitness = 0
Iteration 26: Best Fitness = 0
Iteration 27: Best Fitness = 0
Iteration 28: Best Fitness = 0
Iteration 29: Best Fitness = 0
Iteration 30: Best Fitness = 0
Iteration 31: Best Fitness = 0
Iteration 32: Best Fitness = 0
Iteration 33: Best Fitness = 0
Iteration 34: Best Fitness = 0
Iteration 35: Best Fitness = 0
Iteration 36: Best Fitness = 0
Iteration 37: Best Fitness = 0
Iteration 38: Best Fitness = 0
Iteration 39: Best Fitness = 0
Iteration 40: Best Fitness = 0
Iteration 41: Best Fitness = 0
Iteration 42: Best Fitness = 0
Iteration 43: Best Fitness = 0
Iteration 44: Best Fitness = 0
Iteration 45: Best Fitness = 0
Iteration 46: Best Fitness = 0
Iteration 47: Best Fitness = 0
Iteration 48: Best Fitness = 0
Iteration 49: Best Fitness = 0
Iteration 50: Best Fitness = 0
Iteration 51: Best Fitness = 0
Iteration 52: Best Fitness = 0
Iteration 53: Best Fitness = 0
Iteration 54: Best Fitness = 0
Iteration 55: Best Fitness = 0
Iteration 56: Best Fitness = 0
```

```
Iteration 57: Best Fitness = 0
Iteration 58: Best Fitness = 0
Iteration 59: Best Fitness = 0
Iteration 60: Best Fitness = 0
Iteration 61: Best Fitness = 0
Iteration 62: Best Fitness = 0
Iteration 63: Best Fitness = 0
Iteration 64: Best Fitness = 0
Iteration 65: Best Fitness = 0
Iteration 66: Best Fitness = 0
Iteration 67: Best Fitness = 0
Iteration 68: Best Fitness = 0
Iteration 69: Best Fitness = 0
Iteration 70: Best Fitness = 0
Iteration 71: Best Fitness = 0
Iteration 72: Best Fitness = 0
Iteration 73: Best Fitness = 0
Iteration 74: Best Fitness = 0
Iteration 75: Best Fitness = 0
Iteration 76: Best Fitness = 0
Iteration 77: Best Fitness = 0
Iteration 78: Best Fitness = 0
Iteration 79: Best Fitness = 0
Iteration 80: Best Fitness = 0
Iteration 81: Best Fitness = 0
Iteration 82: Best Fitness = 0
Iteration 83: Best Fitness = 0
Iteration 84: Best Fitness = 0
Iteration 85: Best Fitness = 0
Iteration 86: Best Fitness = 0
Iteration 87: Best Fitness = 0
Iteration 88: Best Fitness = 0
Iteration 89: Best Fitness = 0
Iteration 90: Best Fitness = 0
Iteration 91: Best Fitness = 0
Iteration 92: Best Fitness = 0
Iteration 93: Best Fitness = 0
Iteration 94: Best Fitness = 0
Iteration 95: Best Fitness = 0
Iteration 96: Best Fitness = 0
Iteration 97: Best Fitness = 0
Iteration 98: Best Fitness = 0
Iteration 99: Best Fitness = 0
Iteration 100: Best Fitness = 0
Final solution:['3', '3', '3', '3', '3'] - Final fitness:0
```