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

def objective_function(pipe_sizes, flow_rates, demand, node_pressure, pipe_costs, pump_costs):
    pipe_cost = np.sum(pipe_sizes ** 2 * pipe_costs)
    pump_cost = np.sum(flow_rates * pump_costs)

    pressure_penalty = 0
    for i in range(len(node_pressure)):
        if node_pressure[i] < 20:
            pressure_penalty += (20 - node_pressure[i]) ** 2

    demand_penalty = 0
    for i in range(len(demand)):
        if flow_rates[i] < demand[i]:
            demand_penalty += (demand[i] - flow_rates[i]) ** 2

    total_cost = pipe_cost + pump_cost + pressure_penalty + demand_penalty
    return total_cost

class GreyWolfOptimization:
    def __init__(self, num_wolves, max_iter, demand, pipe_costs, pump_costs, num_nodes):
        self.num_wolves = num_wolves
        self.max_iter = max_iter
        self.demand = demand
        self.pipe_costs = pipe_costs
        self.pump_costs = pump_costs
        self.num_nodes = num_nodes

        self.wolves = np.random.rand(self.num_wolves, 2 * self.num_nodes)

        self.alpha = None
        self.beta = None
        self.delta = None
        self.alpha_score = float('inf')
        self.beta_score = float('inf')
        self.delta_score = float('inf')

    def fitness(self, wolf):
        pipe_sizes = wolf[:self.num_nodes]
        flow_rates = wolf[self.num_nodes:]

        node_pressure = np.random.rand(self.num_nodes) * 50

        return objective_function(pipe_sizes, flow_rates, self.demand, node_pressure, self.pipe_costs, self

    def update_positions(self):
        for i in range(self.num_wolves):
            A = 2 * np.random.rand(1) - 1
            C = 2 * np.random.rand(1)
            D_alpha = np.abs(C * self.alpha - self.wolves[i])
            X1 = self.alpha - A * D_alpha

            A = 2 * np.random.rand(1) - 1
            C = 2 * np.random.rand(1)
            D_beta = np.abs(C * self.beta - self.wolves[i])
            X2 = self.beta - A * D_beta

            A = 2 * np.random.rand(1) - 1
            C = 2 * np.random.rand(1)
            D_delta = np.abs(C * self.delta - self.wolves[i])
            X3 = self.delta - A * D_delta

            self.wolves[i] = (X1 + X2 + X3) / 3

    def optimize(self):
        for _ in range(self.max_iter):
```

```
            for i in range(self.num_wolves):
                fitness_value = self.fitness(self.wolves[i])

                if fitness_value < self.alpha_score:
                    self.alpha_score = fitness_value
                    self.alpha = self.wolves[i]

                elif fitness_value < self.beta_score:
                    self.beta_score = fitness_value
                    self.beta = self.wolves[i]

                elif fitness_value < self.delta_score:
                    self.delta_score = fitness_value
                    self.delta = self.wolves[i]

            self.update_positions()

        return self.alpha

num_wolves = 30
max_iter = 100
num_nodes = 5
demand = np.array([50, 40, 30, 60, 80])
pipe_costs = np.array([1, 2, 1.5, 3, 2])
pump_costs = np.array([0.1, 0.1, 0.1, 0.1, 0.1])

gwo = GreyWolfOptimization(num_wolves, max_iter, demand, pipe_costs, pump_costs, num_nodes)
best_solution = gwo.optimize()

best_pipe_sizes = best_solution[:num_nodes]
best_flow_rates = best_solution[num_nodes:]

print("Best Pipe Sizes:", best_pipe_sizes)
print("Best Flow Rates:", best_flow_rates)
```

```
Best Pipe Sizes: [0.7028334  0.73479142 0.58696454 0.48379806 1.18978869]
Best Flow Rates: [0.8846324  1.0455686  1.11733696 0.81021675 1.38541497]
```