# graywolfoptm-1

November 28, 2024

[6]:
```
#gray wolf optimization for water flow optimization
print("Name:Sudarshan Komar","USN:1BM22CS291",sep="\n")

import numpy as np
import matplotlib.pyplot as plt


def objective_function(pipe_sizes, flow_rates, demand, node_pressure,␣
↪pipe_costs, pump_costs):
    # Pipe costs (cost proportional to diameter^2)
    pipe_cost = np.sum(pipe_sizes ** 2 * pipe_costs)  # cost related to the␣
↪pipe diameters

    # Pumping costs (assume pump power is proportional to flow rate)
    pump_cost = np.sum(flow_rates * pump_costs)  # cost related to pumping

    # Pressure constraints: penalize if pressure falls below threshold (say 20m)
    pressure_penalty = 0
    for i in range(len(node_pressure)):
        if node_pressure[i] < 20:
            pressure_penalty += (20 - node_pressure[i]) ** 2  # Penalize low␣
↪pressure

    # Demand satisfaction: penalize if flow at any node does not meet demand
    demand_penalty = 0
    for i in range(len(demand)):
        if flow_rates[i] < demand[i]:
            demand_penalty += (demand[i] - flow_rates[i]) ** 2  # Penalize␣
↪under-supply

    # Total objective: cost + penalties for pressure and demand violations
    total_cost = pipe_cost + pump_cost + pressure_penalty + demand_penalty
    return total_cost

# Define the Grey Wolf Optimization (GWO) class
class GreyWolfOptimization:
```

```python
    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):
        # Split wolf's position into pipe sizes and flow rates
        pipe_sizes = wolf[:self.num_nodes]  # First half of wolf is for pipe sizes
        flow_rates = wolf[self.num_nodes:]  # Second half of wolf is for flow rates

        # Initialize pressure array (just as an example, in practice you would calculate this based on network model)
        node_pressure = np.random.rand(self.num_nodes) * 50  # Random pressure values for each node (example)

        # Call the objective function to calculate cost
        return objective_function(pipe_sizes, flow_rates, self.demand, node_pressure, self.pipe_costs, self.pump_costs)

    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

            # Update the wolf's position
            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])

                # Update alpha, beta, and delta wolves based on fitness values
                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]

            # Update positions of all wolves
            self.update_positions()

        return self.alpha  # Return the best solution

num_wolves = 30
max_iter = 1000
num_nodes = 6
demand = np.array([50, 100, 80, 150, 120, 180])
pipe_costs = np.array([1.0, 3.5, 1.3, 1.8, 2.0, 1.7])
pump_costs = np.array([0.2, 0.2, 0.18, 0.25, 0.2, 0.2])

# Initialize and run the Grey Wolf Optimization
gwo = GreyWolfOptimization(num_wolves, max_iter, demand, pipe_costs,
 ↪pump_costs, num_nodes)
best_solution = gwo.optimize()

# Extract best solution: pipe sizes and flow rates
best_pipe_sizes = best_solution[:num_nodes]
best_flow_rates = best_solution[num_nodes:]
```

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

```
Name:Sudarshan Komar
USN:1BM22CS291
Best Pipe Sizes: [0.00091585 0.00095113 0.00084839 0.00140677 0.00067415
0.00074783]
Best Flow Rates: [0.00055832 0.00113251 0.00048501 0.00093122 0.00049868
0.00107437]
```