```
import numpy as np

def objective_function(generation):
    cost = 0
    for i in range(len(generation)):
        a_i, b_i, c_i = generation_params[i]
        cost += a_i * generation[i]**2 + b_i * generation[i] + c_i
    return cost

def check_constraints(generation, demand, tolerance=0.01):
    power_balance = np.sum(generation) - demand
    for i in range(len(generation)):
        if generation[i] < gen_min[i] or generation[i] > gen_max[i]:
            return False
    if abs(power_balance) > tolerance * demand:
        return False
    return True

class GreyWolfOptimizer:
    def __init__(self, num_wolves, max_iter, dim, demand):
        self.num_wolves = num_wolves
        self.max_iter = max_iter
        self.dim = dim
        self.demand = demand

        self.positions = np.random.uniform(gen_min, gen_max, (self.num_wolves, self.dim))
        self.best_position = np.copy(self.positions[0])
        self.best_score = float("inf")

    def optimize(self):
        for t in range(self.max_iter):
            for i in range(self.num_wolves):
                generation = self.positions[i]

                if check_constraints(generation, self.demand):
                    score = objective_function(generation)

                    if score < self.best_score:
                        self.best_score = score
                        self.best_position = generation
                else:
                    score = float('inf')

            a = 2 - t * (2 / self.max_iter)
            for i in range(self.num_wolves):
                for j in range(self.dim):
                    r1 = np.random.rand()
                    r2 = np.random.rand()
                    A = 2 * a * r1 - a
                    C = 2 * r2
                    D = np.abs(C * self.best_position[j] - self.positions[i, j])
                    self.positions[i, j] = self.best_position[j] - A * D
                    self.positions[i, j] = np.clip(self.positions[i, j], gen_min[j], gen_max[j])

        return self.best_position, self.best_score

num_generators = 3
generation_params = [(0.1, 10, 500), (0.2, 5, 300), (0.15, 8, 400)]
gen_min = np.array([50, 30, 40])
gen_max = np.array([150, 120, 100])
demand = 300
num_wolves = 30
max_iter = 100
gwo = GreyWolfOptimizer(num_wolves, max_iter, num_generators, demand)
best_generation, best_cost = gwo.optimize()
print("Best Generation Outputs (MW):", best_generation)
print("Total Generation Cost: $", best_cost)
```

```
Best Generation Outputs (MW): [68.56519559 72.02205001 95.75003078]
Total Generation Cost: $ 7818.331295177577
```