IDS 435 - Assignment 4

- First Name: Yelizaveta
- Last Name: Semikina
- UIN: 670246811

Electric Power Generation This assignment is based on the week 12 lecture regarding electric power generation.

Consider the problem description and formulation in the file "electrical_power_1_gcl_formulation.html". We also reviewed accompanying Python code to solve this problem using Gurobi and analyze the output. See "electrical_power_1_gcl_code.ipynb". The assignment requires you to account for new developments regarding available generators. In addition to the thermal generators in the previous problem, two new Hydro-power generators have become available, each with a fixed power output (when on):

| Hydro plant | Output (MW) | | A | 900 | | B | 1400 |

The costs associated with using a hydro plant are slightly different. There's an hourly cost, but it is much smaller than the hourly cost of a thermal generator. The real cost for a hydroelectric plant comes from depletion of the water in the reservoir, which happens at different rates for the two units. The reservoir must be replenished before the end of the time horizon by pumping water into it, which consumes electricity. A hydroelectric plant also has a startup cost.

| Hydro plant | Cost per hour (when on) | Startup cost | Reservoir depth reduction (m/hr) | | A | 90\|1500 | 0.31 | | B | 150\|1200 | 0.47 |

Pumping water into the reservoir consumes electricity at a rate of 3000 MWh of electricity per meter of height. The height of the reservoir at the end of the time horizon must be equal to the height at the beginning.

All the thermal generators should still be considered and the constraints that applied in the problem we saw in class also apply here and the goal is the same: Which thermal and Hydro generators should be committed to meet anticipated demand in order to minimize total cost?

Pumping water into the reservoir consumes electricity at a rate of 3000 MWh of electricity per meter of height. The height of the reservoir at the end of the time horizon must be equal to the height at the beginning.

All the thermal generators should still be considered and the constraints that applied in the problem we saw in class also apply here and the goal is the same: Which thermal and Hydro generators should be committed to meet anticipated demand in order to minimize total cost?

1) Modify the formulation we discussed in class and documented in "electrical_power_1_gcl_formulation.html" to account for the new Hydro generators and its constraints. I have posted the jupyter file associated with the in class version of the formulation so that you can modify that file.

2) Modify the Gurobi implementation to account for the changes in (1) and solve the revised problem and show the optimal plan for which generators should be committed as well as the minimum cost.

Question 1

Decision Variables:

$\text{ngen}_{t,p}$: Number of generators of type t that are on in time period p. This variable shows the choice to activate a particular kind of generator for a given amount of time.

$\text{output}_{t,p}$: Total power output from generators of type t in time period p. This variable shows the actual power reuslt from a specific generator in a specific amount of time.

$\text{nstart}_{t,p}$: Number of generators of type t to start in time period p. This variable shows the how additional specific generators in a specific time period can start to work for specific demand.

Parameters:

period-hours_p: Number of hours for a specific time period. This parameter shows the duration of a specific time period.

generators_t: Number of generators of type t. This parameter shows the total number of generators available for every type.

demand_p: Total power demand for time period p. This parameter shows how much electricty would be required for the specific time period.

start0: Number of generators that are at the beginning of the time horizon. This parameter shows how many generators are already at the start of the time horizon.

min-output_t: Minimum output for generator typet t (when on). This parameter shows the minimum output that a specific type of generator can make.

max-output_t: Maximum output for generator type t. This parameter shows the maximum output that a specific type of generator can make.

base-cost_t: Minimum operating cost (per hour) for a generator of type t. This parameter shows the cost per hour of operating a generator of a specific type at its minimum output.

per-mwh-cost_t: Cost to generate one additional MW (per hour) for a generator of type t. This parameter shows the cost per hour to provide one extra megawatt of power for a specific generator type.

startup-cost_t: Startup cost for generator of type t. This parameter shows the cost of starting a specific type of generator

Objective Function: The objective function minimizes the total cost for electricity demand. The total cost has three components:

1. Z_{on}: Cost of maintaining generators at their lowest output levels for each type and time period.
2. Z_{extra}: Cost of generating more energy than is necessary for each type at time period.
3. $Z_{startup}$: Cost of starting generators in a specific time period.

In [1]:
```
!pip install gurobipy
```

```
Requirement already satisfied: gurobipy in /Users/liza/opt/anaconda3/lib/python
3.9/site-packages (10.0.1)
```

In [2]:
```
"""
_____________________________
Code for Question 2.
_____________________________
"""
import gurobipy as gp
from gurobipy import GRB

# Parameters
ntypes = 3
nperiods = 5
maxstart0 = 5

generators = [12, 10, 5]
period_hours = [6, 3, 6, 3, 6]
demand = [15000, 30000, 25000, 40000, 27000]
min_load = [850, 1250, 1500]
max_load = [2000, 1750, 4000]
base_cost = [1000, 2600, 3000]
per_mw_cost = [2, 1.3, 3]
startup_cost = [2000, 1000, 500]

hydro_plants = ['A', 'B']
hydro_output = {'A': 900, 'B': 1400}
hydro_cost_hourly = {'A': 90, 'B': 150}
hydro_startup_cost = {'A': 1500, 'B': 1200}
reservoir_reduction_rate = {'A': 0.31, 'B': 0.47}
pumping_electricity_rate = 3000

# Create model
model = gp.Model('PowerGeneration')

# Decision Variables
ngen = model.addVars(ntypes, nperiods, vtype=GRB.INTEGER, name="ngen")
nstart = model.addVars(ntypes, nperiods, vtype=GRB.INTEGER, name="nstart")
output = model.addVars(ntypes, nperiods, vtype=GRB.CONTINUOUS, name="genoutput")
hydron = model.addVars(hydro_plants, nperiods, vtype=GRB.BINARY, name="hydron")
ncommit = model.addVars(ntypes, vtype=GRB.INTEGER, name="ncommit")
```

```
gen_types = ['thermal', 'hydro']
n_gen = len(gen_types)

# Constraint 1
numgen = model.addConstrs((ngen[t,p] <= generators[t] for t in range(n_gen) for

# Constraint 2
numgen_commit = model.addConstrs((ngen.sum(t, '*') == ncommit[t] for t in range(
# Constraint 3
# lower bound
min_max_load = model.addConstrs((output[t,p] >= min_load[t] * ncommit[t] for t i
# upper bound
min_max_load = model.addConstrs((output[t,p] <= max_load[t] * ncommit[t] for t i

# Constraint 4
demand_constraint = model.addConstrs((gp.quicksum(output[t,p] for t in range(n_g

# Objective Function
# calculates the total cost of a power system optimization problem
total_cost = gp.quicksum(base_cost[t] * ncommit[t] + per_mw_cost[t] * output[t,p

model.setObjective(total_cost, GRB.MINIMIZE)
model.optimize()


for p in range(nperiods):
    print("Period:", p)
    committed_generators = []
    for t in range(n_gen):
        if ncommit[t].x > 0:
            committed_generators.append(gen_types[t])
            print("\tGenerator Type:", gen_types[t])
            print("\tNumber of Generators Committed:", ncommit[t].x)
            print("\tAssociated Output:", output[t,p].x)
            print("\tAssociated Cost:", base_cost[t] * ncommit[t].x + per_mw_cos
    if 'A' in hydron.keys() and hydron['A', p].x > 0:
        print("\tHydro Plant A Committed")
        print("\tAssociated Output:", hydro_output['A'])
        print("\tAssociated Cost:", hydro_cost_hourly['A'] + (hydro_startup_cost
    if 'B' in hydron.keys() and hydron['B', p].x > 0:
        print("\tHydro Plant B Committed")
        print("\tAssociated Output:", hydro_output['B'])
        print("\tAssociated Cost:", hydro_cost_hourly['B'] + (hydro_startup_cost
    print("Total Cost:", model.objVal)
    print("Committed Generators:", committed_generators)
    print()
```

```
Set parameter Username
Academic license - for non-commercial use only - expires 2024-03-28
Gurobi Optimizer version 10.0.1 build v10.0.1rc0 (mac64[x86])

CPU model: Intel(R) Core(TM) i5-8257U CPU @ 1.40GHz
Thread count: 4 physical cores, 8 logical processors, using up to 8 threads

Optimize a model with 37 rows, 58 columns and 82 nonzeros
Model fingerprint: 0xb98f1425
Variable types: 15 continuous, 43 integer (10 binary)
Coefficient statistics:
  Matrix range     [1e+00, 3e+03]
```

```
  Objective range  [1e+00, 1e+04]
  Bounds range     [1e+00, 1e+00]
  RHS range        [1e+01, 4e+04]
Found heuristic solution: objective 312920.00000
Presolve removed 37 rows and 58 columns
Presolve time: 0.00s
Presolve: All rows and columns removed

Explored 0 nodes (0 simplex iterations) in 0.01 seconds (0.00 work units)
Thread count was 1 (of 8 available processors)

Solution count 1: 312920

Optimal solution found (tolerance 1.00e-04)
Best objective 3.129200000000e+05, best bound 3.129200000000e+05, gap 0.0000%
Period: 0
        Generator Type: thermal
        Number of Generators Committed: 17.0
        Associated Output: 15000.0
        Associated Cost: 47000.0
Total Cost: 312920.0
Committed Generators: ['thermal']

Period: 1
        Generator Type: thermal
        Number of Generators Committed: 17.0
        Associated Output: 24000.0
        Associated Cost: 65000.0
Total Cost: 312920.0
Committed Generators: ['thermal']

Period: 2
        Generator Type: thermal
        Number of Generators Committed: 17.0
        Associated Output: 19000.0
        Associated Cost: 55000.0
Total Cost: 312920.0
Committed Generators: ['thermal']

Period: 3
        Generator Type: thermal
        Number of Generators Committed: 17.0
        Associated Output: 34000.0
        Associated Cost: 85000.0
Total Cost: 312920.0
Committed Generators: ['thermal']

Period: 4
        Generator Type: thermal
        Number of Generators Committed: 17.0
        Associated Output: 21000.0
        Associated Cost: 59000.0
Total Cost: 312920.0
Committed Generators: ['thermal']
```

The solution showed that using thermal generators was the best way to save money. We found out how many generators to use and how much electricity they should make in each period. The total cost was 312920.0

Constraints:

Constraint 1 sets a limit on the number of generators of each type that can be used in each period. Specifically, for each generator type t and period p, it limits the number of generators ngen[t,p] that can be used to be less than or equal to a fixed limit generators[t].

Constraint 2 ensures that the total number of generators of each type that are committed to be used across all periods equals a fixed amount ncommit[t]. In other words, it ensures that the same number of generators are committed to be used throughout the entire time horizon of the model, regardless of how many are actually used in each individual period.

Constraint 3 imposes lower and upper bounds on the output of each generator type at each time period based on their minimum and maximum load capacity, respectively.

Constraint 4 adds a demand constraint that ensures that the total output from all generators plus the pumping electricity rate for hydropower equals the electricity demand at each time period.

A new binary variable hydron and an associated constraint was added to the model to indicate whether the hydropower plants are turned on at each time period.