Chapter 5

Project

5.1 General description

The Unit Commitment problem in energy management aims at finding the optimal production schedule of a set of generation units, while meeting various system-wide constraints. It has always been a large-scale, non-convex, difficult problem, especially in view of the fact that, due to operational requirements, it has to be solved in an unreasonably small time for its size. In this project we will play with this problem a bit. We refer the reader to [9] and references therein for detailed information.

From an abstract perspective we are given a set of assets $i = 1, ..., m$ capable of generating power. These assets consist of thermal plants, cascading reservoir systems, renewable intermittent generators. Each asset comes with an abstract feasible set $X_i \subseteq \mathbb{R}^{n_i}$ assumed closed, cost function $c_i : \mathbb{R}^{n_i} \to \mathbb{R}$ and collection of variables $x_i \in \mathbb{R}^{n_i}$ along with a matrix A_i such that $A_i x_i$ is the amount of power generated by the asset at the various time steps. A vector $d \in \mathbb{R}^T$, with T the total number of time steps, each of length Δt (hours) is the system wide load.

The abstract problem reads:

$$\begin{aligned} \min_{x_1,\ldots,x_m} \quad & \sum_{i=1}^{m} c_i(x_i) \\ \text{s.t.} \quad & x_i \in X_i, \\ & \sum_{i=1}^{m} A_i x_i \geq d. \end{aligned}$$

It is very popular to cast this unit-commitment problem as a MILP. This is by no means the only path, nor necessarily the most desirable one. It is however fitting for this course.

This project will thus consist of a writing down and implementation of the resulting optimization problem as well as a resolution / testing of the code.

5.2 Thermal plants

Consider the situation of a (thermal) power plant producing over a time horizon T at each time step p_t, $t = 1, ..., T$ (MW) of active power. Each time step has a duration Δt in hours. The power plant has a proportional cost c_t in €/ MWh ; Power levels are subject to a minimal and maximum power output level when running $p_t^{\texttt{min}}, p_t^{\texttt{max}}$ respectively. Furthermore the power levels are subject to gradient restrictions, i.e., adjacent power levels should not exceed g_t, where g_t is in MW/h; When the plant is online, it needs to remain so for a minimum of τ_+ time steps. When the plant is offline it needs to remain so for a total of τ_- time steps.

Thermal plants moreover have a startup cost s_t € that has to be paid

5.3 Cascading reservoir systems

A cascading reservoir system can be understood as a directed graph $G = (V, A)$, wherein each vertex $v \in V$ represents a reservoir with initial volume V_0 (m^3), minimal volume $V_t^{\texttt{min}}$ and maximal volume $V_t^{\texttt{max}}$ in m^3. Each reservoir also receives inflows a_t in m^3/s. The arcs are directed from uphill to downhill and represent turbines (going with the direction) or pumps (going against the direction).

Each turbine comes with a flow rate f_t in m^3/s as well as ramping conditions g in $(m^3/s)/h$. The flow rate are subject to bounds $\underline{f}_t, \overline{f}_t$ and moreover we are given the following form of the hydro production function:

$$p(f) = \min_{j \in J} p_j + \langle \rho_j, f - f_j \rangle \,,$$

with for $j \in J$: f_j (m^3/s), p_j (MW) a fixed collection of points. The resulting power output of a turbine should also be subject to bounds $\underline{p}, \overline{p}$.

Each pump likewise comes with a flow rate (negative) but with a unique linear "HPF" : $p(f) = \rho f$. Note that physically pumps require power to pump water uphill. Evidently this must go at an overall loss (think about why?).

5.4 Description of the task at hand

In order to tackle the project, the first step consists of making precise the sets of constraints X_i for the various thermal and hydro assets. To this end observe that i as hydro is concerned relates to a cascading reservoir system as a whole. Once these constraints are written down, write the full UC problem as a MILP: think carefully about the proper formulation.

The next step consists in writing a computer program that implements the model. It will have to read data from some source, populating the model, build the model, solve the model, output the solution to some human readable format.

The proper way to structure the program is to clearly distinguish between the various steps. Data reading and curing should not be done during the building of the model phase. Once the model is built, it should be exportable to some classic format (MPS of LP files for instance), making it solver agnostic. The solving should be done with some solver (CPLEX, Gurobi, or other available), possibly playing with parameters of the solver. The data output phase should provide some display of the solution in an understandable way. In this case, we should at least be able to see how much power is generated by each asset at each time step. For cascading reservoir systems we may wish to see the flow of water in between reservoirs, or how the volumes in each reservoir evolve over time.

In terms of programming language, Python PULP, Pyomo or Julia JUMP are likely good suggestions.

5.5 Data

Download some thermal unit-commitment test data at `https://gitlab.com/smspp/ucblock/-/tree/develop/netCDF_files/UC_Data/T-Ramp`. The netcdf files should be transformed into clear text through ncdump.