**GOOD Model Formulation**

Prior to reading this document, please make sure you are familiar with the mathematical model formulation provided by Alan.

**Overview**

The GOOD Model V2 aims to leverage the object-oriented structure of Python. The V1 model was strictly procedural only using the built-in objects within the pyomo package ([Read the pyomo docs](https://pyomo.readthedocs.io/en/stable/) to learn more). V2 will continue to use pyomo as the optimization package and incorporate objects to make the more flexible and extensible for future needs (herein ‘the model’ shall refer to V2). The model will still utilize pyomo components – sets, parameters, variables, objective, and constraints. Importantly, all pyomo does is write a .lp file that is passed to a solver (i.e., cbc, gurobi, highs, etc.). The solver is the component which finds the optimal solution while pyomo serves as the interface to it. Solvers are either free (cbc, glpk, highs) or require a commercial license (gurobi). As this model intends to be open source, it will use a free solver. If a user has access to a commercial solver, they can easily insert it into the model.

**Structure**

The model structure will reflect a node link graph where the nodes are each region, and the links are the transmission lines connecting the regions. Structuring the model as a graph better reflects the physical architecture of the electric grid and it more easily enables use of Python objects. The model includes eight objects represented as classes:

* Regions (nodes)
* Transmission (links)
* Generators
* Solar
* Wind
* Storage
* Load
* Optimization

Generators, Solar, Wind, Storage, and Load are associated with each region but are given their own classes as they have distinct characteristics. These classes handle the corresponding input data or model parameters, such as generator capacities or cost, and define the decision variables, terms for the objective function, and constraints. Additionally, this construction enables more flexible and extensible modeling for future needs. For instance, if a user wants to model specific managed charging approaches or other demand response strategies, they could do by manipulating the Load class and its methods without having the edit other parts of the model. In each region the pyomo model components (sets, parameters, variables, objective, constraints) are created by the corresponding object – Generator, Solar, Wind, Storage, or Load. For example, the Generator class handles the generator capacity and cost data translating each into parameters, the generation decision variable, the generation term for the objective function, and generation-specific constraints. A similar process is completed for the Link objects.

The optimization object is the class users will directly interact with it. The class parses the data, instantiates the model, and creates the region and transmission objects, and collects the parameters, variables, objectives, and constraints created in the objects into the larger model. Finally, it solves the model. Figure 1 outlines model structure.

Figure 1.

A diagram of a class

Description automatically generated

**Balancing Constraints**

An important component of every electricity system model is an energy balancing constraint – supply and demand must be always equal. The balancing constraint is managed at each the region level and is stored within the Region class. This constraint incorporates the parameters and variables created in the Generator, Solar, Wind, Storage, Load, and Transmission classes to compose the constraint.