This is a demonstration of a Transmission Expansion Planning (TEP), a way of planning the building of an electric infrastructure system. In this kind of TEP, a central planner, such as a nation, decides where to construct new lines based on the goal of maximizing social welfare while minimizing costs, while generation is controlled by motivate private companies. Social welfare in this scenario is defined as meeting all electrical demand of the population consistently and as economically as possible, and costs include both yearly line construction costs and hourly generation outlays on generation. While research into efficient solutions of the deterministic version of TEP have been ongoing for decades, introducing probabilistic constraints can greatly increase the complexity and create intractable models, even on the most modern computer hardware. One way to address the problem in some reasonable amount of time while also including the extra complexity of parameter uncertainty to use robust optimization, which can be reduce the complexity to solve a problem, but runs the risk of over-emphasizing worst-case scenarios [1]

Implementation will be on Pyomo (www.pyomo.org/), an optimization modeling language written in Python [2][3]. Since Pyomo is an open source software, and most other work on robust TEPs has used algebraic modelling software that require users to purchase a license, there is hopefully demand for this development. If not, well it’s here anyway ¯_(ツ)_/¯. Resources about Pyomo that may be helpful:

Documentation: https://pyomo.readthedocs.io/en/stable/

Git Repository: https://github.com/Pyomo

Installation

To run a user needs to have Python and Pyomo solved (www.pyomo.org/). Pyomo works by calling an external Mixed Integer Linear Programming (MILP) solver which also must be downloaded. Experimental results provided here are found using the CPLEX solver from IBM (www.cplex.com). Another option is Gurobi (www.gurobi.com/). However, these both require the purchase of a license if you are a non-academic user. An option with a free software license is the GLPK solver from the GNU project (www.gnu.org/software/glpk/) However, this list is not exhaustive, and reference the Pyomo documentation for all possibilities.

Note this software is still very much in development and should not be used by anyone, living or dead, for any purpose.

Tests

These provide a series of tests to establish the validity of the program. Note, that development is in process and several tests were done on versions of the program using different sets of parameters. See

github.com/Homersmyid/TEP/blob/master/Tests/Readme.pdf

for a description of the tests. Each provides a self-contained code that can be run to repeat the test.

Files

RuizMain.py

The program to run. In this case implements the constraint-and-column method from Ruiz and Conjego [4]. Additional resources were provided by [1], in the chapter on TEP problems. As of now contains a constant that denotes the initial setup of the program.

Start_x_star – A list of possible connections between nodes. Each tuple contains three numbers representing the starting node, ending node, and number of lines on connection. 0 represents no connection.

Stop – Number of iterations to stop after. All examples in the literature have finished in 10 iterations or less.

RuizC.py

The file RuizC.py contains constants to use for the program.

Solver – Pyomo calls an external solver to solve a MILP. As such, the user can decide any solver they have installed. Experimental results are found using the CPLEX solver from IBM (www.cplex.com). However, to use this requires to purchase a license if you are a non-academic user. An option with a free software license is the GLPK solver from the GNU project (www.gnu.org/software/glpk/)

Epsilon – The relative distance between the upper bound and lower bound found for the optimal solution to stop the algorithm at:

$$(Upper - Lower)/_{Upper}$$

So, for example with an epsilon of $1e^{-6}$ and upper bound of $2e^{-8}$, the algorithm will stop if the lower bound is found to be within 200. A standard is $1e^{-6}$.

Data – The location of the data file. Assumed to be an AMPL style data file. See the documentation of Data Files to see how to set up the file.

Unctol – A tolerance to use on deciding if the data file has the same upper bounds as lower bounds on any set of uncertain parameters. This is to avoid a divide by zero situation.

MIPGAP – If using the CPLEX solver, the relative gap to stop the mixed integer solver at. See CPLEX documentation for more information

References

[1] Antonio J. Conejo, Luis Baringo Morales, S. Jalal Kazempour, and Afzal S. Siddiqui. 2016. Investment in Electricity Generation and Transmission: Decision Making Under Uncertainty (1st ed.). Springer Publishing Company, Incorporated.

[2] Hart, William E., Carl D. Laird, Jean-Paul Watson, David L. Woodruff, Gabriel A. Hackebeil, Bethany L. Nicholson, and John D. Siirola. *Pyomo – Optimization Modeling in Python*. Second Edition. Vol. 67. Springer, 2017.

[3] Hart, William E, and David L Woodruff. “Pyomo: Modeling and Solving Mathematical Programs in Python.” *Mathematical Programming Computation*, vol. 3, no. 3, 2011, pp. 219–260.

[4] [3] Ruiz, C, and A.J Conejo. “Robust Transmission Expansion Planning.” *European Journal of Operational Research*, vol. 242, no. 2, 2015, pp. 390–401.