Author: *Xingpeng Li*, initially documented in 2016, last updated in April 2023.
Website: https://rpglab.github.io/people/Xingpeng-Li/
Email: xplipower@gmail.com

SCED User Manual

In today's power system operations, security-constrained economic dispatch (SCED) is one of the key functions in the energy management system (EMS). In general, SCED is an optimization process that aims to provide the least cost generation that meets all the operation and reliability constraints.

Typically, SCED is a linear programming (LP) problem since all generators' status are considered as fixed input and DC power flow model is used. Though AC power flow model is more accurate, it is not used due to its computational complexity. The non-linearity and non-convexity may cause divergence of the SCED problem and hence AC power flow model based SCED is not stable. On the contrary, DC power flow model based SCED is a LP problem, which is much simpler and the optimal solution can be guaranteed if it is feasible.

In terms of time-frame, the two types of SCED applications are day-ahead (DA) SCED and real-time (RT) SCED. DA-SCED considers multiple periods while RT-SCED only consider one-period. In this manual, SCED is for RT-SCED unless indicated specifically.

RT-SCED calculates the desired MW output level for all on-line units repeatedly with updated information. For instance, RT SCED runs every 5 minutes for PJM. The results from RT SCED are the actual dispatch signals that will be sent to each generator.

In terms of input data type, there are deterministic SCED and stochastic SCED. For deterministic SCED, all inputs, including load profile and generation of renewable energy resources, are assumed to be accurate without any forecasting error. However, for stochastic SCED, some degrees of uncertainty can be taken into account. As shown in Fig. 1, it is observed that the most possible value of the wind power is 100 MW, which is the only input for deterministic SCED; however, for stochastic SCED, the input can be a set of several discrete points (80 MW, 90 MW, 100 MW, 110 MW, and 120 MW) with weights assigned to them, or a range from 80 MW to 120 MW. Available mathematical methods include robust programming, stochastic programming, and chance programming. This set of codes is created for deterministic SCED only.

The SCED is expected to use the same model and function the same way with the PJM RT SCED. However, due to data unavailability and unknown tricks insider the PJM's commercial SCED, it is impossible to have exactly the same results. The input to this SCED tool includes the

system network topology, generator status, load forecast, and forecasting generations of renewable energy resource. Its objective is to minimize the total cost, subject to physical constraints and other requirements such as reserve requirements. The output would be the dispatch point of each generator.

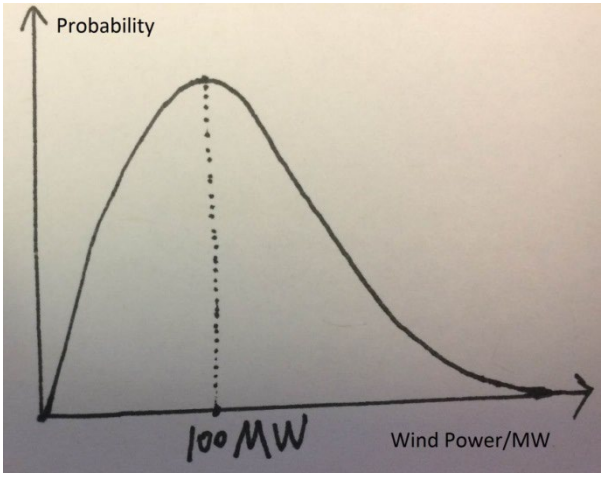

Fig. 1 Probability density function curve

The SCED can handle large-scale systems and serves as a simulator for off-line study. It is also able to load different input data formats. There are two sets of data for the PJM system. One is referred to as real-case and the other is referred to as generic-case. Real-case is a set of data files containing EMS information and market information. Generic-case is a simplified version of real-case, which contains much less data and hidden all sensitive information. Therefore, the SCED for generic-case is referred to as G-SCED while the SCED for real-case is referred to as R-SCED.

1 Pyomo set up

Pyomo is an open-source optimization library based on python. Before starting to install pyomo, users should make sure python is installed in their computers. Some available *Scientific Python distributions* can be downloaded from the following link:

http://www.scipy.org/install.html

Install pyomo with Anaconda (one of the scientific Python distributions):

Open “Anaconda Prompt”, type “pip install pyomo”

The following operations may not be needed if you have a scientific Python distributions installed in your computer.

In "Anaconda Prompt", type "pip install numpy"

In "Anaconda Prompt", type "pip install scipy"

In "Anaconda Prompt", type "pip install pyomo.extras"

Reference: http://www.pyomo.org/installation/

How to test pyomo is installed successfully in your computer?

Open cmd, if it is not in C:\ disk, type "C:", press "Enter". Then, type "pyomo -h"; you will see some related information if pyomo is installed correctly.

2 Third party solver set up

Two open-source free solvers that have been tested are: glpk and cbc. Note that glpk 4.60 does not work while glpk 4.55 works. The latest cbc 2.7.5 can work. To call these two solvers, users can add their directory in the windows system environment variable PATH. This may require administrator rights. Note that after the system variable PATH is set for solvers, users may have to restart the computers to make the solver work.

For glpk: the directory containing the *glpsol executable* (glpsol.exe) must be in the list of paths defined by the PATH environment variable. GLPK-4.55 for windows can be downloaded from the below link. You need to unzip it. https://sourceforge.net/projects/winglpk/files/winglpk/

For cbc: the directory containing the *cbc executable* (cbc.exe)must be in the list of paths defined by the PATH environment variable. Cbc-2.7.5 for windows can be downloaded from the below link. You need to unzip it. http://www.coin-or.org/download/binary/Cbc/

There could be newer version of both glpk and cbc available. However, test them before trust them. At least, glpk-4.60 does not work very well with Pyomo-4.3.11328.

3 Configure file

All basic input parameters are specified in the file with a name of "configure.txt". In this file, any contents after "//" or "#" in the same line will be ignored.

//---------- Solver settings ----------//

@solverName: name of the solver.

@solverTimLimit: maximum computational time allowed for solver, in seconds.

@solverOptGap: optimization relative gap; if it is reached, solver will stop and exit.

//---------- SCED settings ----------//

@isRunSCED: if true, SCED simulator will run; if false, no SCED will be run.

@isPyomoDataFilesAvailable:

This parameter will be used ONLY when @isRunSCED is set to TRUE.

If true, SCED will directly load the input data file as specified by the parameters

@pathRealCase + @pyomoDataFormatInputFileRC for real-case, or

@pathGenericCase+ @pyomoDataFormatInputFileGCfor generic-case.

If false, the program will first generate those files from initial/original real-case or generic-case data files.

@generatePyomoDataFiles:

This parameter will be used ONLY when @isRunSCED is set to FALSE.

True means the program will produce pyomo-format based data files while 'false' will generate normal format based files.

@needHeading:

This parameter will matter only when the code is generating regular (non-pyomo) format data files. If true, first line is heading.

@runSCEDTimeFrame: SCED will be run every 'runSCEDTimeFrame' (most likely 5) minutes. This parameter is only used for determining the SCED period with the input time.

@blockPrice: Its unit is $, this parameter will matter ONLY when the code is generating pyomo input data file. It is used to linearize the slope cost curve.

@isPositivePgPmaxPminNeeded: if true, pg, pgmax, pgmin will be set to zero if they are negative.

@handle_CostCurveSegment_Pgmin: if true, the program will automatically fix the conflict between cost_curve_based_Pgmax and pgmin (cost_curve_based_Pgmax < pgmin).

@costCurveSegment_Pgmin_option: this parameter will matter only when CostCurveSegment is set to true. If it is 1, pgmin will be changed to cost_curve_based_Pgmax.

//---------- Date and Time ----------//

@Year, @Month, @Day, @Hour, @Minute, @Second:

These six parameters are used for the real-case study. Since some original data files for real-case are used for an entire day, while the pyomo-format case input file corresponds to a specific time slot, so this date and time information are needed for updated information search and match.

//---------- Real case input data ----------//

@pyomoDataFormatInputFileRC: input real-case that will be sent to pyomo. This parameter has to be in the same directory with the program codes.

@pathRealCase: directory of the original files for a real-case

@bidFileNameRC: daily bid data

@costCurveRampFileNameRC: costcurve ramping rate

@hourlyDataFileNameRC: hourly updated unit data

@interfaceFileNameRC: daily interface/constraint data

@reserveReqFileNameRC: reserve requirement

@aolFileNameRC: AOL file containing the dFAX (PTDF) data

@rawFileNameRC: PSS/E power flow raw file.

@rtUnitStatusFileNameRC: updated unit status data

@unitFileNameRC: This csv file is used to match unit name and unit ID; one line heading.

//---------- Generic case input data ----------//

@pyomoDataFormatInputFileGC: input generic-case that will be sent to pyomo. This parameter has to be in the same directory with the program codes.

@pathGenericCase: directory of the original files for a generic-case.

@costCurveOutputFileNameGC: unit cost curve data.

@costCurveMultiRampFileNameGC: unit energy ramping rate data.

@costCurveSpinRampFileNameGC: unit spinning ramping rate data.

@rawFileNameGC: PSS/E power flow raw file.

4 Main programs

There are two main programs:

- RunSCEDGenericCaseModel.py
- RunSCEDRealCaseModel.py.

Assume the programs are in the directory of C:/Users/obama/sced/. To run SCED simulation for generic case, open "cmd", change its current working directory to C: disk by typing "C:" and then press "Enter" key. Then, use command "cd" to enter each directory until the inside of sced folder; for instance, in "C:\", type "cd Users" will move the working directory to "C:\User".

When you are in the directory of C:/Users/obama/sced/, type "python RunSCEDGenericCaseModel", press "Enter", then, the G-SCED starts to run. If you need to all the information that shown in the screen saved to a file, use this command "python RunSCEDGenericCaseModel > filename.txt". So, if there is already a file that has the same name of "filename.txt", the program will first delete that file, and creates a new file with the same name. If you want to keep the old file, and still need to save those information to the same file, use command "python RunSCEDGenericCaseModel >> filename.txt"; then, all output will be appended to that file.

You can use the same procedure of G-SCED to run R-SCED. Just remember to change the name from "RunSCEDGenericCaseModel" to "RunSCEDRealCaseModel".

Another way to run the simulation is to use *Spyder's GUI run function*.

5 Log file

Each of the two main programs will automatically generate a log file with date and time in its name. From this log file, the user can check how much time each specific part takes and, then, determine which block of code is the performance bottleneck.

For SCED application on a large-scale system like the PJM system, the bottleneck would be pyomo itself rather than the solver or the developer's code.

6 G-SCED

G-SCED is a SCED for the generic case. The generic case for SCED consists of 4 data files. They are a power flow (PSS/E raw) file, an energy ramping rate file, a spinning ramping rate file, and a unit cost curve file.

There is no unit commitment in G-SCED model, which means that it will only not change the status of any unit. G-SCED only re-dispatch the generators that have a cost curve available, and fixes the outputs of all other generators. Energy ramping rate is used to restrict the change in generator output between the starting time and ending time of the same period. Spinning ramping rate is used to restrict the generator output change for the situations when a contingency occurs. *N*-1 is not represented explicitly; the largest generator contingency is used to as the spinning reserve requirement. Specific contingency constraints can be specified in the input data files for G-SCED. Slack variables are used to make sure the problem is feasible.

7 R-SCED

R-SCED is a SCED for the real-case. The real case for SCED consists of 9 data files. They are listed below.

- power flow (PSS/E raw) file
- bid data (unit bid data, operation related data as well as market related data)
- cost curve and ramping rate file (contain information of energy ramping rate, spinning ramping rate, and cost curve)
- Hourly unit data: updated unit data for each operational hour. If contradiction (e.g. different unit economic maximum output) exists between bid data file and this file, use this one.
- Interface limit (thermal constraints)
- local_as_service (reserve requirements)
- AOL file (contains the dFAX data)
- Rt unit status fix (contain information of the unit schedule status). “U” means that the associated schedule is not available.
- Units.csv (converted from units.xls file). It is used to match unit name and unit ID.

Hourly unit data file will provide unit regulation qualification status and spinning reserve qualification status (not all units are qualified for providing regulation reserve or spinning reserve service), regulation offer MW, spinning reserve offer MW, hourly updated economic maximum output and economic minimum output.

Similar to G-SCED, there is no unit commitment in R-SCED model, which means that it will only not change the status of any unit. R-SCED only re-dispatch the generators that have a cost curve available, and fixes the outputs of all other generators. Energy ramping rate is used to restrict the change in generator output between the starting time and ending time of the same period. Spinning ramping rate is used to restrict the generator output change for the situations when a contingency occurs. Some *N*-1 contingency constraints are represented explicitly. Slack variables are used to make sure the problem is feasible.

8 Outputs

One data file generated by the "run" program is "*genData_Actually.txt*". It contains the generator data that are used by the solver.

The file "WriteResults.py" is a subroutine that is to dump desired results to files on disk. If there are files with the same name on disk; then, they will be deleted first and then be created by the code.

results_summary.txt shows the objective values, load shedding, and all non-zero slack variables.

results_load.txt lists all the load data, which is actually used by the solver. Load may be adjusted by a simple factor to represent the system loss.

9 Others

Note that, if you do not have a specific set of data, for instance, no interface limit, then, just do not have any data related to interface limit in the input data file; just like no such type of constraint modeled in the problem.

10 References

[1] Xingpeng Li and Kory W. Hedman, "Enhanced Energy Management System with Corrective Transmission Switching Strategy— Part I: Methodology," *IEEE Transactions on Power Systems*, vol. 34, no. 6, pp. 4490-4502, Nov. 2019.

[2] Xingpeng Li and Kory W. Hedman, "Enhanced Energy Management System with Corrective Transmission Switching Strategy— Part II: Results and Discussion," *IEEE Transactions on Power Systems*, vol. 34, no. 6, pp. 4503-4513, Nov. 2019.