SynGrid 1.0
User's Manual

Zhifang Wang Hamidreza Sadeghian Seyyed Hamid Elyas
Ray D. Zimmerman Eran Schweitzer Anna Scaglione

December 18, 2018

Contents

List of Figures

List of Tables

1 Introduction

1.1 Background

SynGrid is a package of MATLAB language M-files[1] for automatically creating synthetic power grid models for use with MATPOWER. The SynGrid project page can be found at:

https://github.com/MATPOWER/mx-syngrid

SynGrid is based on code written by Dr. Zhifang Wang and her group, including Seyyed Hamid Elyas and Hamidreza Sadeghian, at Virginia Commonwealth University, and further developed as part of the "Synthetic Data for Power Grid R & D" project[2] under the ARPA-E GRID DATA program,[3] in conjunction with Ray Zimmerman from Cornell University and Eran Schweitzer and Anna Scaglione from Arizona State University. SynGrid builds upon previous work [1, 2] by Wang, Scaglione, and Thomas and other recent works [3–7, 9, 10].

[1]Also compatible with GNU Octave [11].
[2]https://arpa-e.energy.gov/?q=slick-sheet-project/synthetic-data-power-grid-rd
[3]https://arpa-e.energy.gov/?q=arpa-e-programs/grid-data

1.2 License and Terms of Use

The code in SynGrid is distributed under the 3-clause BSD license [12]. The full text of the license can be found in the `LICENSE` file at the top level of the distribution or at https://github.com/MATPOWER/mx-syngrid/blob/master/LICENSE and reads as follows.

1.3 Citing SynGrid

While not required by the terms of the license, we do request that publications derived from the use of SynGrid explicitly acknowledge that fact by citing one or more of references [2], [5] and [6], namely:

> Z. Wang, R. J. Thomas, and A. Scaglione, "Generating Statistically Correct Random Topologies for Testing Smart Grid Communication and Control Networks", *Smart Grid, IEEE Transactions on*, vol. 1, no. 1, pp. 28–39, June 2010.
> DOI 10.1109/TSG.2010.2044814
>
> S. H. Elyas, Z. Wang, "Improved Synthetic Power Grid Modeling with Correlated Bus Type Assignments", *Power Systems, IEEE Transactions on*, vol. 32, no. 5, pp. 3391–3402, Sept. 2017.
> DOI 10.1109/TPWRS.2016.2634318
>
> E. Schweitzer, A. Scaglione, "A Mathematical Programing Solution for Automatic Generation of Synthetic Power Flow Cases," *Power Systems, IEEE Transactions on*, 2018.
> DOI 10.1109/TPWRS.2018.2863266

1.4 SynGrid Development

The SynGrid project is based on an open development paradigm, hosted on the SynGrid GitHub project page:

`https://github.com/MATPOWER/mx-syngrid`

The SynGrid GitHub project hosts the public Git code repository as well as a public issue tracker for handling bug reports, patches, and other issues and contributions. There are separate GitHub hosted repositories and issue trackers for Matpower, SynGrid and the testing framework used by both of them, MP-Test, all available from `https://github.com/MATPOWER/`.

2 Getting Started

2.1 System Requirements

To use SynGrid 1.0 you will need:

- Matlab® version 7.3 (R2006b) or later[4], or

[4] Matlab is available from The MathWorks, Inc. (`http://www.mathworks.com/`). Matlab is a registered trademark of The MathWorks, Inc.

- GNU Octave version 4.2 or later[5]
- MATPOWER 7 or later[6]
- MP-Test (included with MATPOWER), for running the SynGrid test suite.[7]

For the hardware requirements, please refer to the system requirements for the version of MATLAB[8] or Octave that you are using.

In this manual, references to MATLAB usually apply to Octave as well.

2.2 Installation

SynGrid is included in the MATPOWER Extras package[9]. If you have installed MATPOWER 7 or later from the distribution available at the main MATPOWER web site, you should already have SynGrid installed.

If, on the other hand, MATPOWER was installed without SynGrid or the MATPOWER Extras (e.g. directly from the GitHub repository), SynGrid can be installed manually, either by installing MATPOWER Extras (according to the instructions at `https://github.com/MATPOWER/matpower-extras`) or by following the directions below.

Installation and use of SynGrid requires familiarity with the basic operation of MATLAB or Octave, including setting up your MATLAB path.

Step 1: Clone the repository or download and extract the zip file of the SynGrid distribution from the SynGrid project page[10] to the location of your choice. The files in the resulting `mx-syngrid` or `mx-syngridXXX` directory, where `XXX` depends on the version of SynGrid, should not need to be modified, so it is recommended that they be kept separate from your own code. We will use <*SYNGRID*> to denote the path to this directory.

Step 2: Add the following directories to your MATLAB or Octave path:[11]

[5]GNU Octave [11] is free software, available online at `http://www.gnu.org/software/octave/`.

[6]MATPOWER is available at `http://www.pserc.cornell.edu/matpower/`

[7]MP-Test is also available at `https://github.com/MATPOWER/mptest`.

[8]`http://www.mathworks.com/support/sysreq/previous_releases.html`

[9]`https://github.com/MATPOWER/matpower-extras`

[10]`https://github.com/MATPOWER/mx-syngrid`

[11]As an alternative to manually modifying your MATLAB or Octave path, you can simply move/rename the directory at <*SYNGRID*> so it resides at <*MATPOWER*>`/extras/syngrid`, and then re-run `install_matpower`.

- <*SYNGRID*>`/lib` – core SynGrid classes and functions
- <*SYNGRID*>`/lib/t` – test scripts for SynGrid

Step 3: At the MATLAB prompt, type `test_syngrid` to run the test suite and verify that SynGrid is properly installed and functioning.[12] The result should resemble the following:

```
>> test_syngrid
t_sg_options.........ok
t_sgvm_add_shunts....ok
t_sgvm_data2mpc......ok
t_syngrid............ok
t_syngrid_vm.........ok
All tests successful (174 of 174)
Elapsed time 173.16 seconds.
```

2.3 Creating a Synthetic Grid Model

Running SynGrid to create a synthetic power system model, a MATPOWER case, involves a simple call to the `syngrid` function, specifying the desired number of buses in the resulting model.

2.3.1 Running SynGrid

For example, to generate a 200-bus model, simply type:

```
mpc = syngrid(200);
```

The result is a MATPOWER case struct that can be passed directly to a power flow or optimal power flow solver.

```
pf_results = rundcpf(mpc);
opf_results = rundcopf(mpc);
```

It can also be saved directly to a MATPOWER case file by specifying a file name in the optional third input argument.

```
mpc = syngrid(200, sgopt, 'case_200sg');
```

[12]The tests require functioning installations of MATPOWER and MP-Test (included with MATPOWER).

2.3.2 Setting Options

The optional second input argument to the `syngrid` function is used to specify SynGrid options. This argument is a struct, described in detail in Table 6-3 in Chapter 6. The `sg_options` function is used to construct a default options struct or set or modify the options. For example, the following example specifies the level of progress output and the loading level.

```
sgopt = sg_options('verbose', 1, 'bm.loading', 'H');
mpc = syngrid(200, sgopt);
```

2.4 Documentation

There are two primary sources of documentation for SynGrid. The first is this manual [13], which gives an overview of the capabilities and structure of SynGrid and describes the methods used to create the synthetic grids. This manual can be found in your SynGrid distribution at <*SYNGRID*>`/docs/SynGrid-manual.pdf` and the latest version is always available at: `https://github.com/MATPOWER/mx-syngrid/blob/master/SynGrid-manual.pdf`.

The second source of documentation is the built-in `help` command. As with the built-in functions and toolbox routines in MATLAB and Octave, you can type `help` followed by the name of a command or M-file to get help on that particular function. All of the M-files in SynGrid have such documentation and this should be considered the main reference for the calling options for each function.

3 Overview of SynGrid

SynGrid has two primary modes of operation, referred to as *base* mode and *variations* mode.

Given only the desired network size, the base mode automatically generates a single power grid test case from scratch and returns or saves it in the standard Matpower case format. The generated model features the same kind of small-world "electric" topology and statistical properties as found in realistic power grids, and has all of the data necessary to run DC power flow (PF) and DC optimal power flow (OPF), including electric topology, bus types, generation capacities, load and generation settings, transmission line capacities, and generation cost models.

The variations method creates multiple AC OPF cases from "seed" first order data, as illustrated conceptually in Figure 3-1. This method samples branch data, w_ℓ containing line ratings, impedances etc., and bus data, w_n containing load and generation limits, and places them onto a topology. In essence, this approach exploits OPF solutions to progressively alter how the bus and branch property samples are placed onto a topology, with the goal of achieving the desirable power flow behavior.

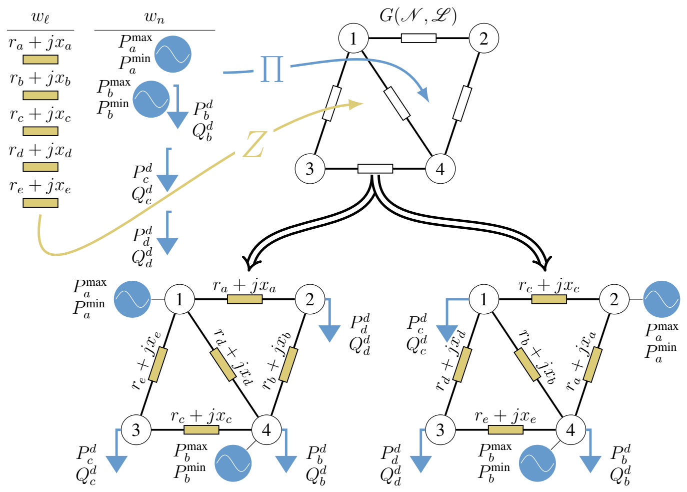


Figure 3-1: Conceptual illustration of how the SynGrid variations mode operates. The main computational challenge is solving for permutations Z and Π.

4 SynGrid – Base Mode

The base mode synthetic grid generation begins with the topology creation using small-world properties and realistic power grid features. The second step is bus type assignment. It utilizes a direct search procedure for the best generation, load and connection bus assignment with examining a numerical measure, called "bus type entropy". Once the bus types are determined, generation capacities, generation dispatch and load settings are assigned in the next step. The algorithms utilize non-trivial correlations between electric parameters to assign statistically random set of variables in each step. By providing all necessary data for DC PF study such as electric topology, generation and load settings, DC PF is executed to test the state variables including phase angle differences and flow distribution. If the created case does not pass the predefined criteria for the state variables, it will be sent to the previous modules to modify the created synthetic case. Once the generated case passed the state variables test, the transmission line capacity module determines the line capacities, based on the synthetic grid flow distributions. The module for generation cost modelling uses the generation capacities to determine the fuel types and then assigns the cost models based on them.

The SynGrid base mode functionality currently consists of the five components described in the sections below.

4.1 Generation of Electrical Topology

This module generates random grid-unique electric topologies with scalable network size featuring the same kind of small-world electrical topology of real world power transmission network. Based on the fact that in real world power grids usually a large-scale systems consist of a number of smaller-size subsystems, which are inter-connected by sparse and important tie lines, SynGrid base mode generates the cases in a hierarchical way. First, it forms connected subnetworks with size limited by the connectivity requirements. Then connects the subnetworks through lattice connections and finally, generates the line impedances from specific distributions and assign them to the links in the topology network.

The first step to form subnetworks is to select the size of the them according to connectivity limitation. To produce a sparse but connected topology, the small-world requires $\langle k \rangle << N << e^{\langle k \rangle}$. On the other hand real-world power grids have a very low average nodal degree with $\langle k \rangle = 2 \sim 5$, regardless of the network size. Therefore, this limits the network size much smaller than 150 in order to produce a connected topology. Then, to link selection of subnetworks, a number k of links at random will be selected from a local neighborhood N_{d_0} ($N^i_{d_0} = \{j; |j-i| < d_0$ for node i) with the distance threshold of d_0, where k comes from a geometric distribution. Final step to form subnetworks is link rewiring. A Markov chain with transition probabilities of (α, β) is used to select clusters of nods and therefore groups of links to be rewired and the by a specific rewiring probability $q_r w$ which is directly obtained from statistical estimates, some links will be selected to rewire. For the lattice connections, an integer number around $\langle k \rangle$ will be selected at random from neighboring subnetworks to form the whole large-scale power grid network. A number M of line impedances will be generated from a heavy-trailed distributions, and then sorted by magnitude and group into local links, rewire links, and lattice connections. More details are available in [2].

4.2 Bus Type Assignment

The best set of generation, load and connection bus assignments for a given synthetic power grid will be located in this component. There exist non-trivial correlations between the three bus types and other topology metrics such as node degrees and clustering coefficients in a real-world power grid; and a random permutation of the grid's original bus type assignments will dramatically change the grid dynamics and makes the resulting network have no longer like a power grid. Therefore, the Bus Type Entropy has been defined as a numerical weighted measure to quantify and characterize the "correlated" bus type assignments. Two different mathematical definitions for Bus Type Entropy are as following:

$$W_0(\mathbb{T}) = -\sum_{k=1}^{3} r_k \cdot \log(r_k) - \sum_{k=1}^{6} R_k \cdot \log(R_k), \tag{4.1}$$

$$W_1(\mathbb{T}) = -\sum_{k=1}^{3} \log(r_k) \cdot N_k - \sum_{k=1}^{6} \log(R_k) \cdot M_k, \tag{4.2}$$

where $r_k = \frac{N_k}{N}$ represent the bus type ratio where $r_1 = \frac{N_1}{N}$, $r_2 = \frac{N_2}{N}$ and $r_3 = \frac{N_3}{N}$ are the ratios of generation (G), load (L) and connection (C) buses in a grid with network size N, respectively. And $R_k = \frac{M_k}{M}$, corresponding to the link type ratio where R_{1_6} represent the ratios of the six types of branches in a grid, i.e. GG, LL, CC, GL, GC, LC, respectively, with M_k being the total number of branches of a specific link type in the grid.

The best set of bus type assignments will be determine in four steps. First, the empirical PDF of randomized bus type assignments with respect to the grid size and its connecting topology will be generate to calculate the estimated fitting distribution parameters of (μ, σ). Then, the scaling property of normalized distance called d will be calculated using following equations

$$d_{W_0}(N) = \begin{cases} -1.721 \cdot \ln(N) + 8 & \ln(N) \leq 8 \\ -6.003 \times 10^{-14} \cdot (\ln(N))^{15.48} & \ln(N) > 8, \end{cases} \tag{4.3}$$

$$d_{W_1}(N) = \begin{cases} -1.748 \cdot \ln(N) + 8.276 & \ln(N) \leq 8 \\ -6.053 \times 10^{-22} \cdot (\ln(N))^{24.10} & \ln(N) > 8, \end{cases} \tag{4.4}$$

Once the fitting parameters and d are calculated, the target entropy value of W^* can be calculate using $W^* = \mu + \sigma \cdot d_W(N)$, which is consistent with that observed in realistic grids. Finally, an optimization algorithm with objective function of

$$\min_{\forall \mathbb{T}} \varepsilon = |W_1(\mathbb{T}) - W^*| \tag{4.5}$$

will be implemented to search for the desired bus type assignments with respect to the W^* to determine the best set of bus type assignments. More details on bus type assignment can be found in [5].

4.3 Generation Capacities, Load and Generation Settings

Phases three and four generate statistically correct random set of generation capacities and loads and assign them to the generation and load buses in a synthetic grid. Algorithms are according to the estimated distribution of individual generation capacities and loads, approximated scaling function of aggregated generation capacity and load and the correlation between generation capacity and loads with nodal degree of related buses. In real-world power grids, more than 99% of the generation units/loads follow an exponential distribution with about 1% having extremely large capacities falling out of the normal range defined by the expected exponential distribution. Therefore, a statistically correct random set of generation capacities and load will generate based on the derived exponential distribution distributions. For both aggregate generation capacity and load the approximate scaling function can be represented as function of network size as

$$\log_{g,Max}^{tot}(N) = -0.21 \cdot \log(N)^2 + 2.06 \cdot \log(N) + 0.66, \tag{4.6}$$

$$\log_{L}^{tot}(N) = -0.20 \cdot \log(N)^2 + 1.98 \cdot \log(N) + 0.58, \tag{4.7}$$

where $P_{g,Max}^{tot}$ denotes the estimated total generation capacity, P_L^{tot} denotes the estimated total demand, and the logarithm is with base 10. The generated random sets of generation capacities and loads will be checked and scaled to ensure the aggregated generation capacities and loads remain in the range specified by Eqs. 4.6 and 4.7. The final step is to assign the generated statistically correct random sets to the related buses. There exists a non-trivial correlation between the normalized nodal degree of a generation bus and its capacity by a Pearson coefficient of $\rho(\overline{P_{g_n}^{Max}}, \overline{k_n}) \in [0.15, 0.50]$. Moreover, for the normalized node degree of load buses and their loads, the Pearson's coefficient is $\rho(\overline{P_{L_n}}, \overline{k_n}) \in [0.30, 0.60]$. Based on these non-trivial correlations, empirical 2-D probability mass function (PMF) are extracted to use as a guidance map to assign generated random sets of generation capacities and loads based on the node degrees. Details on generation capacity and load assignment are available in [8].

Generation dispatch is the other vital component to build a valid synthetic grid. It enables the toolkit to perform DC PF study, test the created synthetic cases, and use the calculated flow distributions to determine transmission line capacities. Our statistical studies indicate that there exist non-trivial correlation between the generation capacities and its generations dispatch as evidenced by the Pearson coefficient of $\rho(\overline{P_{g_n}^{Max}}, \overline{P_{g_n}}) \in [0.75, 0.95]$ evaluated for realistic power grid data. All the generators in the system can be divided to three categories based on their dispatch factor: I) uncommitted unites with $\alpha_g = 0$, II) partially committed unites with $\alpha_g < 1$, and III) fully committed units with $\alpha_g \approx 1$. It is found that in a typical system about $10 \sim 20\%$ of generators are uncommitted with small and medium generation capacity and about $40 \sim 50\%$ of the generators belong to category II. There exist a significant correlation between the normalized generation capacities and their dispatch factor $(\alpha = \frac{P_g}{P_g^{max}})$ with a Pearson coefficient $\rho(\overline{P_{g_n}^{Max}}, \overline{\alpha_n}) \in [0.15, 0.55]$ in realistic power grids. Therefore, we can extract 2-D PMF for the given reference realistic grid. To reproduce a similar correlation between the generation dispatch and the generation capacity as found in real grid systems, we formulate a 2-D table based on 2-D empirical PMF and assign dispatch factors to each generation bus based on generation capacities and 2-D PMF table.

The next step is to generate statistically correct random set of dispatch factors. We found that for realistic power grids, more than 99% of the committed generations have capacities following an exponential distribution with about 1% having extremely large capacities falling outside of the normal range defined by expected exponential distribution indicated by imperial probability density function (PDF) committed unites. Given a synthetic grid topology with N buses and determined generation capacities and load settings, the algorithm selects a randomly correct set of generation capacities for both committed and uncommitted units which follows 99% rule for committed unites and uniform distribution for uncommitted unites. Next by generating statistically correct random set of dispatch factors, it assigns them to the selected generation capacities in generation buses based on obtained statistical pattern.

Given a synthetic grid topology with N buses and determined generation capacities and load settings, the algorithm selects a randomly correct set of generation capacities for both committed and uncommitted units which follows 99% rule for committed unites and uniform distribution for uncommitted unites. Next by generating statistically correct random set of dispatch factors, it assigns them to the selected generation capacities in generation buses based on a 2-D empirical PMF of normalized generation capacities and dispatch factors. More details are available in [9].

4.4 Transmission Line Capacity

Our initial experiments on statistical distribution of transmission gauge ratio show that the best distribution, which fits the distribution of transmission gauge ratios $\beta_l = \frac{F_l}{F_l^{Max}}$ is exponential distribution. The statistics collected from realistic grids also indicate that there exists a considerable correlation between the transmission gauge ratio and its flow distribution. The Pearson's coefficient varies in range of 0.35 – 0.65. Studying correlation between transmission gauge ratio and normalized flow distribution, we may extract an empirical 2-D PMF and based on that a 2-D PMF table can be formulated to enable an algorithm to assign transmission line capacities to the synthetic power grid. It is worthy to note that in power systems, three different capacities define for transmission lines i.e. long term, short term, and emergency capacity and in this study, we focus on long-term capacity of transmission lines. The transmission line capacities will be determined in three steps. First, a statistically correct random set of transmission gauge ratios will be generated; and using derived 2-D probability distribution table, the transmission gauge ratios will be assigned to each transmission line with respect to the grid flow distribution calculated from the DC power flow solution. More details are available in [9].

4.5 Generation Cost Function

The essential component to perform energy economic studies, such as OPF problem, is to determine generator cost models and their associated coefficients. The algorithm utilizes the statistical analysis of actual grids and available information on generation units to determine fuel types and generation cost models based on pre-determined generation capacities.

The generation mix of U.S. Energy Information Association data is studied to proposed the fuel type assignment algorithm. For simplicity, all different technologies/fuel types in the generation mix are combined into five major categories of Hydro, Wind, Natural Gas, Coal, and Nuclear to avoid relatively complicated modeling for very small portion of total installed generation capacity. Statistical analysis on power plants for all the three North American electric power interconnections shows that there exist a strong correlation between normalized generation capacities of power plants and their discretized fuel types by a Pearson coefficient of $\rho(\overline{P_{g_n}^{Max}}, \overline{TP_n}) \in [0.75, 0.95]$. Given a data set of generation unit capacity and discretized fuel types, we may define a joint distribution function in the two-dimensional space of $(\overline{P_{g_n}^{Max}}, \overline{TP_n})$. The 2-D density function $f(\overline{P_{g_n}^{Max}}, \overline{TP_n})$, when integrated over a set S gives the probability that $(\overline{P_{g_n}^{Max}}, \overline{TP_n})$ falls into the set:

$$\Pr(A) = \Pr\{(\overline{P_{g_n}^{Max}}, \overline{TP_n}) \in S\}, \tag{4.8}$$

Thus, the 2-D empirical probability mass function (PMF) cab be extracted to use as a guidance map to assign fuel types to generators based on their normalized generation capacity. Figure 2 shows imperial 2-D PMF of normalized generation capacity versus discretized fuel types for EI system. Once the fuel types determined, generation cost models will be assigned accordingly. For the generation cost model, we consider two approaches to assign no load and production cost to each generator:

$$\text{Approach } A : c(p) = a_0 + c_f(b_1 P + b_2 P^2), \tag{4.9}$$

$$\text{Approach } B : c(p) = a_0 + a_1 P + a_2 P^2 \tag{4.10}$$

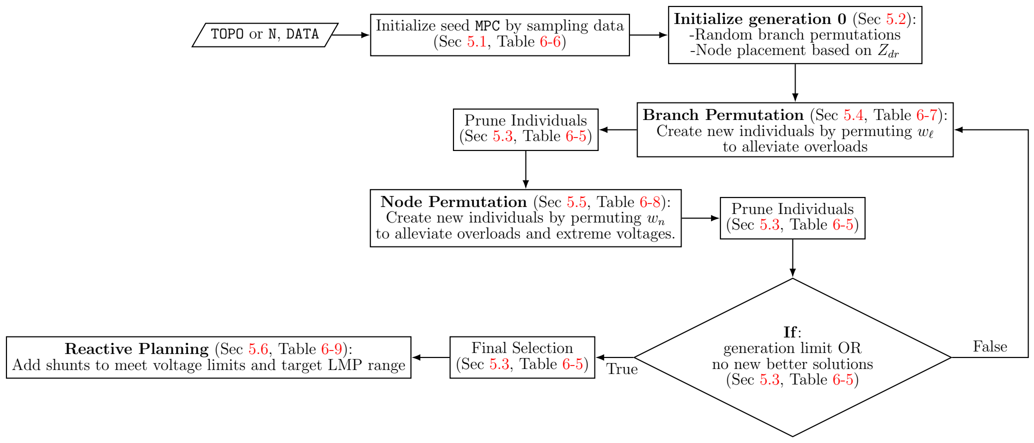
where $\{a_i\colon i = 0, 1, 2\}$ and $\{b_i\colon i = 1, 2\}$ indicate the fuel-dependent cost model coefficients, c_f refers to fuel cost and P is generator output. The approa A utilizes average heat rates of power plants and their fuel costs obtained from EIA data, to model cost functions. On the other hand, A models the generation cost based on the dispatch cost coefficients derived from generation block-offer schedule data of ISO-NE and PJM differentiated by fuel type. No load costs for wind and hydro power plants are set to zero. The cost coefficients data summarized in Table 4-1 and 4-2 are used to assign for each generator by its fuel type and capacity.

Table 4-1: Coefficients Approach 1

Fuel Type	Capacity (MW)	a_0 (\$/h)	c_f (\$/MBtu)	b_1 (MBtu/MWh)	b_2 (MBtu/MWh^2)
Coal	0 - 75	0 - 238	2.16	9.43-18.53	0
Coal	75 - 150	238 - 745	2.16	9.43-18.53	0
Coal	150 - 350	745 - 1213	2.16	9.43-18.53	0
Coal	0 >350	1213 - 3043	2.16	9.43-18.53	0
Natural Gas	0 - 400	0 - 600	2.59	6.5 - 17.5	0
Natural Gas	0 >400	600 - 3859	2.59	6.5 - 17.5	0
Nuclear	- - -	1000 - 1500	0.85	10.46	0
Hydro/Wind	- - -	0	0	0	0

Table 4-2: Coefficients Approach 2

Fuel Type	Capacity (MW)	a_0 (\$/h)	a_1 (\$/MWh)	a_2 (\$/MWh^2)
Coal	0 - 75	0 - 238	19	0.001
Coal	75 - 150	238 - 745	19	0.001
Coal	150 - 35	745 - 1213	19	0.001
Coal	0 >350	1213 - 3043	19	0.001
Natural Gas	0 - 400	0 - 238	23.13 - 57.03	0.002 - 0.008
Natural Gas	0 >400	0 - 238	23.13 - 57.03	0.002 - 0.008
Nuclear	- - -	1000 - 1500	5 - 11	0.00015 - 0.00023
Hydro/Wind	- - -	0	0	0



TOPO or N, DATA
Initialize seed MPC by sampling data (Sec 5.1, Table 6-6)
Initialize generation 0 (Sec 5.2): -Random branch permutations -Node placement based on Z_{dr}
Branch Permutation (Sec 5.4, Table 6-7): Create new individuals by permuting w_ℓ to alleviate overloads
Prune Individuals (Sec 5.3, Table 6-5)
Node Permutation (Sec 5.5, Table 6-8): Create new individuals by permuting w_n to alleviate overloads and extreme voltages.
Prune Individuals (Sec 5.3, Table 6-5)
If: generation limit OR no new better solutions (Sec 5.3, Table 6-5)
False
True
Final Selection (Sec 5.3, Table 6-5)
Reactive Planning (Sec 5.6, Table 6-9): Add shunts to meet voltage limits and target LMP range

Figure 5-1: Variations Mode Flowchart

5 SynGrid – Variations Mode

Figure 5-1 provides a flowchart of the variations mode operation. The following subsections describe the algorithms implemented in each of the subprocesses.

5.1 Creating Seed Data

As illustrated in Figure 3-1 there are two main categories to sample: branch and node properties.

5.1.1 Branch Properties

If the `'direct'` method of sampling is selected then elements are picked uniformly at random from the list provided in the `data` input struct. When the `'kde'` method is selected the process proceeds in three steps.

First, the number of transformers is estimated as:

$$n_x = \text{round}\left[n_l \times \frac{n_x^s}{n_l^s}\right], \tag{5.1}$$

where n_x^s is the number of transformers in the `data` struct, n_l^s the total number of branches in the seed `data` struct. The total branch and transformer samples needed are n_l and n_x. For example, there could be 3000 branch samples in the `data` struct ($n_l^s = 3000$), however, the `topo` input calls for only 1000 samples ($n_l = 1000$) to be created.

In the second step a KDE is fit to all branches that are *not* transformers. The $n_l - n_x$ lines are then sampled from the KDE. Finally, another KDE is fit to all the transformers in `data`, and sampled n_x times.

5.1.2 Bus Properties

The bus properties are somewhat more complicated since both load and generation are needed. Load values can be either sampled directly or using a KDE in a method quite similar to the previous section. Generation values are always sampled directly following the load sampling.

The basic idea is that there are a few different types of configurations are considered:

- Buses with no load and no generation
- Buses with no load and generation
- Buses with both load and generation
- Buses with load only

The number of buses with generators is either calculated based on the ratio in the input data *or* specified directly by the user via the `ngbuses` option. Similarly for the number of no load buses. The number of no load, no generation buses is chosen uniformly at random as between 0.6 and 0.95 of the number of no load buses. With these numbers, the rest of can be trivially derived.

The one additional consideration is regarding number of generators on a bus. If the generator bus is specified in the input data it is used to group generator sets and thus multiple ones can be sampled at once.

Finally, the ratio between generation capacity and load is considered. This is calculated directly from the input data when `usegen2load=1` (default). If `usegen2load=0` then the ratio is sampled uniformly at random between 1.3 and 1.6. Single generators are resampled randomly to try and shift the ratio towards the desired goal. If the `'direct'` sampling method is used, individual loads will also be resampled.

5.2 Initial Permutation

Initial permutation of branch parameters is done uniformly at random. The initial node permutation leverages observations in [18] correlating generators and the driving point impedances, Z_{dr} at their bus. In general generators appear to be located at buses with *larger* magnitude Z_{dr}. Within generator buses, however, *larger* generators are paired with *smaller* Z_{dr}.

To select generator buses, the driving point impedances are ordered in descending order $|Z_{\mathrm{dr}}^1| \geq |Z_{\mathrm{dr}}^2| \geq \ldots \geq |Z_{\mathrm{dr}}^{n_b}|$. The fact that the order statistics of the Uniform distribution are distributed as $\mathrm{Beta}(k, n-k+1)$, with n the number of samples, is used to sample buses "around" a certain range in the order Z_{dr} values. If $n = n_b$, the distribution quickly becomes far too narrow. Therefore $n = 10$ is fixed and the resulting samples are scaled. Specifically, the $\mathrm{Beta}(8, 3)$ distribution is sampled n_g times. Each sample, $s \in [0, 1]$ maps to an index i as:

$$i = \mathrm{round}(s \times n_b). \tag{5.2}$$

In this way n_g indices are selected that are scattered more heavily towards larger Z_{dr} values.

Generators are assigned to the selected buses with negative correlation, so that the largest generator gets the smallest Z_{dr}. Finally, 10% of the generators are randomly selected and swapped with another randomly selected generator to add some noise and avoid perfect negative correlation.

5.3 Evolutionary Algorithm Loop

The main loop of the variations mode is an Evolutionary Algorithm (EA) inspired procedure. Each loop is referred to as a generation, and each generation is composed of several individuals. Individuals in this case are simply MATPOWER cases with different permutations of sets w_ℓ and w_n.

The "evolution" refers to the fact that in each an individual produces a new individual, i.e. MATPOWER case, by permuting some of its properties to achieve some objective. Branch and node properties are handled separately, and those procedures are described in the subsequent sections. Finally, each permutation block can be iterated several times, increasing the number of cases that can be permuted. This is illustrated in Figure 5-2 for the case where both branch and node permutations are performed twice. As such, each generation is in a sense comprised of a few 'sub-generations'. To avoid a dimensional explosion of cases, pruning procedures reduce the number of progenitors (i.e. potential parents) that are passed between permutations blocks.

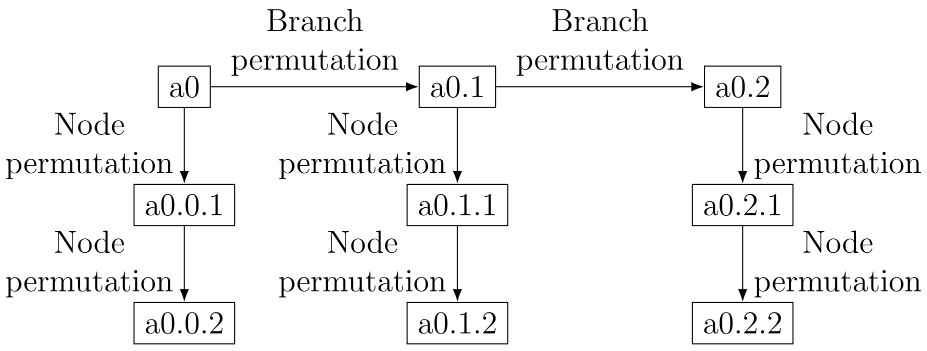


Figure 5-2: Example evolutions from a single parent node 'a0' when both branch and node permutations are performed twice.

5.4 Branch Permutation

5.4.1 General Formulation

The branch permutation problem can be though of as a search for permutation Z of the branch ratings $F_{\max}$ and impedances (r_s and x_s), that would minimize overload flows. In the formulation, flows f are assumed to remain constant. While this is clearly not the case, by minimizing the difference between the old and new impedances, flows are encouraged to remain similar. The problem can be formulated as:

$$\begin{aligned} \underset{t,t_r,t_x,Z}{\textbf{Minimize}} \quad & \mathbf{1}^T(t+t_r+t_x) \\ \textbf{Subject to} \quad & t+(ZF_{\max}-f)\geq 0 \\ & -t_r \leq Zr_s - r_s \leq t_r \\ & -t_x \leq Zx_s - x_s \leq t_x \\ & Z\mathbf{1}=\mathbf{1},;\mathbf{1}^TZ=\mathbf{1}^T \\ & t,t_r,t_x \geq 0 \\ & Z\in\{0,1\} \end{aligned} \tag{5.3}$$

Constraint $t+(ZF_{\max}-f)\geq 0$ in conjunction with $t\geq 0$ forces t_i to be the difference between flow and rating, if f_i is greater than rating $[ZF_{\max}]_i$, and thus discourages overloads. The constraints with r_s and x_s penalize deviation from the original (non-permuted) impedance, and the following constraint forces Z to be a permutation matrix, that is, doubly stochastic and binary valued.

Unfortunately, this problem suffers greatly from dimensionality, since Z is an $n_l \times n_l$ matrix. Therefore, the algorithm in the SynGrid implementation uses a simplified greedy approach to solve for Z. It is important to note that since a greedy approach is used, optimality (even improvement) is not guaranteed.

5.4.2 Simplified Approach

First, the set $\mathcal{B}$ of all overloaded branches that require attention is identified:

$$\mathcal{B} = \left\{ i : \frac{|f_i|}{F^i_{\max}} > x \quad \forall i \in \mathcal{I}_l \right\}, \tag{5.4}$$

where f_i and $F^i_{\max}$ are the flow and rating of branch i, and x is an acceptable ratio. Initially x might be 1 to find only truly overloaded branches. However, it can be useful to allow x to dip below 1, which has the effect of seeking capacity margins on all the branches.

For each element $i \in \mathcal{B}$ there is a candidate set $\mathcal{C}_i$ of possible branches with which to swap properties,

$$\mathcal{C}_i = \left\{ j : F^j_{\max} x \geq |f_i|, |f_j| \leq F^i_{\max} x \quad \forall i \in \mathcal{I}_l \right\}. \tag{5.5}$$

That is, the set of branches whose ratings are *large* enough to support the flow on branch i, and whose flows are *small* enough to be supported by branch i's rating.

The remaining task is to select one candidate, $j^\star$, out of $\mathcal{C}_i$ to swap with branch i. Three tests are used to determine the quality of each candidate:

Impedance Test Measures the distance between the impedance of each branch $j \in \mathcal{C}_i$ and i:

$$z_i(j) = (r^j_s - r^i_s)^2 + (x^j_s - x^i_s)^2. \tag{5.6}$$

Rating Test A Seeks the largest margin between the flow on branch i and the ratings on branches $j \in \mathcal{C}_i$:

$$m^a_i(j) = |f_i| - F^j_{\max} \tag{5.7}$$

Rating Test B Seeks the largest margin between the flow on each branch $j \in \mathcal{C}_i$ and the rating of branch i:

$$m^b_i(j) = |f_j| - F^i_{\max} \tag{5.8}$$

Lower values for all three tests are desirable and therefore, the final selection can be formalized as:

$$j^\star = \arg\min_{j \in \mathcal{C}_i} w_z(j) z_i(j) + w_a(j) m^a_i(j) + w_b(j) m^b_i(j), \tag{5.9}$$

where w_z, w_a, and w_b are used to weights to the different tests.

Letting $\text{ord}_t(j)$ return the order statistic of element $j \in \mathcal{C}_i$ for test t, so that if $t(j)$ is the smallest $\text{ord}_t(j) = 1$, second smallest $\text{ord}(j) = 2$ etc., then the implemented weighting scheme is:

$$w_z(j) = \frac{\text{ord}_{z_i}(j)}{z_i(j)}, \tag{5.10}$$

and similarly for $w_a(j)$ and $w_b(j)$.

5.5 Node Permutation

The node permutation problem is solved in two steps:

1. A desired injection change, ΔP and ΔQ is sought, which, given several assumptions, should minimize overloads and stabilize the voltage profile.
2. A permutation vector π is sought that best achieves the desired change in injection calculated in Step 1.

5.5.1 Desired Injection Change

Bus and branch effects of changing injections, ΔP and ΔQ, are captures by linearizing the power flow around the given operating point. Voltage effects are related via the columns of the system Jacobian associated with the voltage magnitude,

$$\begin{pmatrix} \Delta P \\ \Delta Q \end{pmatrix} = J_v \Delta v, \tag{5.11}$$

and the branch flow effects are captured using AC-PTDFs,

$$\begin{pmatrix} \Delta P_f \\ \Delta Q_f \end{pmatrix} = H_f \begin{pmatrix} \Delta P \\ \Delta Q \end{pmatrix}, \tag{5.12}$$

where Δv, is the change in voltage magnitude, and ΔP_f and ΔQ_f are changes in real and reactive line flows[13].

To create linear constraints on branch flows, the angle of flow on line $i \in \mathcal{I}_l$,

$$\alpha_i = \begin{cases} \arctan\left(\frac{Q_f^i}{P_f^i}\right) & P_f^i \geq 0 \\ \pi - \arctan\left(\frac{Q_f^i}{P_f^i}\right) & P_f^i < 0, \end{cases} \tag{5.13}$$

[13]For simplicity only flows at the *from* end are used in the following

is considered. New flows are only allowed to vary inside the box defined by the intersection points of the rays with angle α_i and $-\alpha_i$ and the limit circle of radius r_i as shown in Figure 5-3. The effective limits are thus,

$$P^i_{f,\max} = r_i \cos(\alpha_i), \quad Q^i_{f,\max} = r_i \sin(\alpha_i). \tag{5.14}$$

With, $P_{f,0}$, $Q_{f,0}$, and v_0 as the initial real and reactive flows, and voltage magnitude, the desired injection change is formulated as the solution to:

$$\underset{\Delta P, \Delta Q, \Delta v}{\textbf{Minimize}} \quad \|\Delta P\|_1 + \|\Delta Q\|_1 + w_p^T s_p + w_q^T s_q + w_v \cdot s_v \tag{5.15a}$$

$$\textbf{Subject to} \quad \begin{pmatrix} \Delta P \\ \Delta Q \end{pmatrix} - J_v \Delta v = 0 \tag{5.15b}$$

$$v_{\min} - v_0 - s_v \le \Delta v \le v_{\max} - v_0 + s_v \tag{5.15c}$$

$$-\begin{pmatrix} P_{f,\max} + s_p \\ Q_{f,\max} + s_q \end{pmatrix} \le \begin{pmatrix} P_{f,0} \\ Q_{f,0} \end{pmatrix} + H_f \begin{pmatrix} \Delta P \\ \Delta Q \end{pmatrix} \le \begin{pmatrix} P_{f,\max} + s_p \\ Q_{f,\max} + s_q \end{pmatrix} \tag{5.15d}$$

$$\mathbf{1}^T \Delta P = 0, \quad \mathbf{1}^T \Delta Q = 0 \tag{5.15e}$$

$$-\Delta P_{\max} \le \Delta P \le \Delta P_{\max}, \quad -\Delta Q_{\max} \le \Delta Q \le \Delta Q_{\max} \tag{5.15f}$$

Constraint (5.15e) forces the net change to be zero, since in the end a permutation is sought, and the limits in (5.15f) are simply the range of the initial injections vector. Additionally, slack variables are added to allow feasibility even in the event that some violations cannot be avoided.

Finally, the desired new nodal injections, $P^\star$ and $Q^\star$ are defined as:

$$\begin{pmatrix} P^\star \\ Q^\star \end{pmatrix} \triangleq \begin{pmatrix} P_0 \\ Q_0 \end{pmatrix} + \begin{pmatrix} \Delta P \\ \Delta Q \end{pmatrix}, \tag{5.16}$$

where, P_0 and Q_0 are the initial injection vectors. The next section seeks a permutation to achieve the desired injections.

5.5.2 Greedy Permutation Approach

The "errors" in real and reactive injection are defined as,

$$\begin{aligned} \epsilon_p &= P^\star - P_0 \\ \epsilon_q &= Q^\star - Q_0, \end{aligned} \tag{5.17}$$

and the total magnitude error is,

$$\epsilon = \sqrt{\epsilon_p^2 - \epsilon_q^2}. \tag{5.18}$$

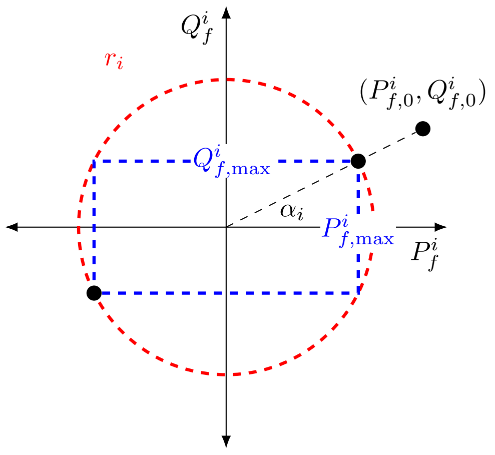


Figure 5-3: Geometrical depiction of how line limits are set during the node permutation procedure.

The greedy node permutation approach fixes each error sequentially, beginning with the largest. That is, vectors $P^\star$, P_0, $Q^\star$, and Q_0 are sorted such that $\epsilon_1 \geq \epsilon_2 \geq \ldots \geq \epsilon_{n_b}$. Algorithm 1 returns the desired permutation vector π. Iterating over all buses, the nearest injection to the desired injection at bus i is found in set x. The selected bus is then removed from the set so that the final result will be a true permutation.

Algorithm 1 Greedy Node Permutation

```
procedure GREEDY NODE PERMUTE($P^\star$, $P_0$, $Q^\star$, $Q_0$)
    $x \leftarrow \{1, 2, \ldots, n_b\}$
    for $i \in \{1, 2, \ldots, n_b\}$ do
        $\pi(i) \leftarrow \arg\min_{j \in x} |P^{\star,i} - P_0^j| + |Q^{\star,i} - Q_0^j|$
        $x \leftarrow x \setminus \{\pi(i)\}$
    end for
    return $\pi$
end procedure
```

5.6 Reactive Planning

The basic reactive planning approach adopted is similar conceptually to [16] but leverages the OPF framework with soft limit capabilities. Generator with limits equal to zero are added to each bus, with soft limits enabled. Since the soft limits are expensive, they will be non-zero only when necessary to satisfy the voltage constraints. Once a solution is found, the desired shunt to add to node i is:

$$B_{sh}^{i} = \frac{s_{\text{qmax}}^{i} - s_{\text{qmin}}^{i}}{|v_i|^2}, \tag{5.19}$$

where s_{qmax}^{i} and s_{qmin}^{i} are violations in the positive and negative directions, respectively, of the soft limit. Note that if the units on s_{qmax}^{i} and s_{qmin}^{i} are MVAr, and v_i is in per-unit, then B_{sh}^{i} will be in MVAr, which is the required format for Matpower.

It is additionally possible to place an upper bound on the soft limits, so that the magnitude of the shunts will be limited. Furthermore, to ensure that no shunt is unreasonably small, all shunts with magnitudes greater than zero but smaller than a given threshold, `tmag`, are set to that threshold:

$$B_{sh}^{i} = \begin{cases} B_{sh}^{i} & (|B_{sh}^{i}| \geq \texttt{tmag}) \cup (B_{sh}^{i} = 0) \\ \operatorname{sgn}(B_{sh}^{i}) \cdot \texttt{tmag} & 0 < |B_{sh}^{i}| < \texttt{tmag}. \end{cases} \tag{5.20}$$

6 syngrid

The functionality of SynGrid is implemented using the `syngrid` function, with an extremely simple interface. For base mode operations, the `syngrid` function takes a single required input argument, the number `N` of desired buses in the case. It returns a standard MATPOWER case struct containing the synthetic case that was created.

```
>> mpc = syngrid(N);
```

Two additional optional input arguments may be used to specify a set of SynGrid options (`sgopt`) and the name (`fname`) of a file to which the resulting MATPOWER case should be saved.

```
>> mpc = syngrid(N, sgopt);
>> mpc = syngrid(N, sgopt, fname);
```

The variations mode, on the other hand, returns a cell array of MATPOWER cases, and requires that the user provide both an existing topology and the data to be sampled and placed on that topology. The topology `topo` can be provided either as a full MATPOWER case (name or struct) or as an $n_l \times 2$ edge matrix of “from” and “to” node indices. Alternatively, using a scalar `N` indicates that a new topology with N nodes should be generated as in base mode. Similarly, the parameters to be sampled are provided in the second argument, `data`, which can also be a full MATPOWER case (name or struct), or a struct with the format described in Table 6-2.

For example, the following uses the topology from the IEEE 118-bus case with data sampled from the ACTIVSg2000 case.

```
>> mpc_array = syngrid('case118', 'case_ACTIVSg2000');
```

And this example, uses a newly created 200 bus topology insteady of the IEEE 118-bus case.

```
>> mpc_array = syngrid(200, 'case_ACTIVSg2000');
```

In variations mode, `syngrid` can also return an optional cell array of status information. It can also take the two optional input arguments `sgopt` and `fname` as in base mode.

The helper function `sgvm_mpc2data` takes a MATPOWER case and extracts the topology matrix `topo` and/or the data for sampling (`data`). So the two calls to `syngrid` in the following example are equivalent.

```
>> mpc_topo = load('case118');
>> mpc_data = load('case_ACTIVSg2000');
>> [~, topo] = sgvm_mpc2data(mpc_topo);
>> data      = sgvm_mpc2data(mpc_data);
>> [mpc_array, status] = syngrid(mpc_topo, mpc_data);
>> [mpc_array, status] = syngrid(topo, data);
```

This could potentially be useful if some modifications are desired before passing the data to SynGrid.

The full set of calling options for `syngrid` in base mode are:

```
>> mpc = syngrid(N);
>> mpc = syngrid(N, sgopt);
>> mpc = syngrid(N, sgopt, fname);
```

And in variations mode:

```
>> mpc_array = syngrid(N, data);
>> mpc_array = syngrid(N, data, sgopt);
>> mpc_array = syngrid(N, data, sgopt, fname);
>> mpc_array = syngrid(topo, data);
>> mpc_array = syngrid(topo, data, sgopt);
>> mpc_array = syngrid(topo, data, sgopt, fname);
>> [mpc_array, status] = syngrid(...);
```

The input and output arguments for both modes are summarized in Table 6-1, with details on the `data` argument in Table 6-2.

Table 6-1: SynGrid Input/Output Arguments

name	description
Input	
`N`	number of buses desired for synthetic case
`topo`†	existing topology, MATPOWER case (struct or name) *or* $n_l \times 2$ topology edge list ("from" and "to" node indices)
`data`†	parameters to sample from, i.e. "seed" information, MATPOWER case (struct or name) *or* struct with fields defined in Table 6-2
`sgopt`	(optional) SynGrid options struct (see Table 6-3)
`fname`	(optional) name of file to which synthetic case will be saved *Note: case is not saved to a file by default.*
Output	
`mpc`	resulting synthetic case as MATPOWER case struct
`mpc_array`†	$1 \times n$ cell array of resulting synthetic cases (MATPOWER case structs), where n is determined by the `vm.ea.select` option
`status`†	$1 \times n$ vector of status values corresponding to the cases returned in `mpc_array` 0 – AC OPF failed to converge 1 – AC OPF solution has possible voltage limit violations, but no branch flow limit violations but 2 – AC OPF solution has no voltage are branch flow limit violations

† Variations mode only.

Table 6-2: Fields of SynGrid `data` Input Argument

field name		description
`baseMVA`†		MVA base for per-unit data (default = 100 MVA)
`branch`	*option 1:*	$n_l \times 6$ matrix corresponding to MATPOWER `branch` matrix columns `BR_R`, `BR_X`, `BR_B`, `RATE_A`, `TAP`, `SHIFT`
	option 2:	struct with $n_l \times 1$ vector fields `'BR_R'`, `'BR_X'`, `'BR_B'`, `'RATE_A'`, `'TAP'`, `'SHIFT'`
`load`	*option 1:*	$n_b \times 2$ matrix corresponding to MATPOWER `bus` matrix columns `PD`, `QD`
	option 2:	struct with $n_b \times 1$ vector fields `'PD'`, `'QD'`
`gen`	*option 1:*	$n_g \times 5$ matrix corresponding to MATPOWER `gen` matrix columns `GEN_BUS`, `QMAX`, `QMIN`, `PMAX`, `PMIN`
	option 2:	$n_g \times 4$ matrix, same as option 1, but without `GEN_BUS` column
	option 3:	struct with $n_g \times 1$ vector fields `'QMAX'`, `'QMIN'`, `'PMAX'`, `'PMIN'`, and optionally `'GEN_BUS'`
`gencost`†		standard MATPOWER `gencost` matrix with n_g rows (default is uniform linear cost for all generators based on `vm.smpl.lincost` option)

† Optional.

The SynGrid options are specified in a struct, typically created and modified by the `sg_options` function. The available options and their default values are summarized in Table 6-3.

Table 6-3: SynGrid Options

name	default	description
`verbose`	0	controls the level of progress output displayed 0 – print no progress info 1 – print a little progress info 2 – print a lot of progress info 3 – print all progress info
`mpopt`	see `mpoption`	MATPOWER options struct (with default AC OPF solver set to `'IPOPT'`)
`mpoptprint`	0	control of MATPOWER printing options 0 – MATPOWER printing turned off 1 – MATPOWER printing controlled by `mpopt`
`bm`		options for base mode
`loading`	`'D'`	loading level `'D'` – default – depends on system size (based on stats from realistic grids) `'L'` – low – total load = 30%-40% of generation capacity `'M'` – medium – total load = 50%-60% of generation capacity `'H'` – high – total load = 70%-80% of generation capacity
`br2b_ratio`	1.5	ratio of number of branches to number of buses (1.25–2.5)
`bta_method`	0	bus type assignment method 0 – "W0" method - simpler & faster 1 – "W1" method - more accurate & slower
`cost_model`	2	generation cost model 1 – linear cost function 2 – quadratic cost function
`br_overload`	0	overloaded lines 0 – without overloaded branches 1 – with same percentage of overloaded lines as realistic grids
`vm`		options for variations mode
`parallel`	see Table 6-4	parallelization options
`ea`	see Table 6-5	evolutionary algorithm options
`smpl`	see Table 6-6	options for sampling input `data`
`branchperm`	see Table 6-7	branch permutation options
`nodeperm`	see Table 6-8	node permutation options
`shunts`	see Table 6-9	reactive planning options

Table 6-4: Parallelization Options (`vm.parallel`)

name	default	description
`use`	0	use `parfor`† for node and branch permutation loops if `use` = 1
`numcores`	0	number of cores to use for parallelization

† Requires MATLAB's Parallel Computing Toolbox.

Table 6-5: Evolutionary Algorithm Options (`vm.ea`)

name	default	description
`generations`	5	number of generations
`inds`	4	number of individuals (cases) in each generation
`randnew`	0	number of new individuals created in a new generation based on random placement rather than as descendants of an individual from the previous generation
`select`	5	number of cases to return upon completion
`initfill`	0	individuals are initialized if `initfill` = 1 and there are fewer than `inds` remaining after a pruning stage

Table 6-6: Sampling Options (`vm.smpl`)

name	default	description
`branch`	`'direct'`	branch sampling options 0, `'none'` – use data exactly as is, ie., no sampling 1, `'direct'` – sample uniformly at random from data 2, `'kde'` – uses a multidimensional KDE fit to the data to generate samples
`node`	`'kde'`	node sampling options 0, `'none'` – use data exactly as is, ie., no sampling 1, `'direct'` – sample load uniformly at random from data 2, `'kde'` – uses a multidimensional KDE fit to the load data to generate samples.
`lincost`	100	default marginal generation cost, used if no `gencost` data is provided
`usegenbus`	1	handling of `GEN_BUS` data when sampling generation 0 – ignore `GEN_BUS` data 1 – use `GEN_BUS` data (if provided)
`ngbuses`	-1	number of buses with generators -1 – choose by scaling ratio in input data
`usegen2load`	1	method used to determine target ratio of generation capacity to load 0 – use random value between 1.3 and 1.6 (uniformly distributed) 1 – determine based on ration in input data

Table 6-7: Branch Permutation Options (`vm.branchperm`)

name	default	description
`niter`	1	number of iterations of branch permutation algorithm per generation
`verbose`	0	level of logging from individual evolution progress 0 – no logging 1 – some logging
`overload_frac_factor`	0.95	factor to reduce ratio x over iterations (see Section 5.4): $x^{\text{new}} = \texttt{overload\_frac\_factor}\vert x$.

Table 6-8: Node Permutation Options (`vm.nodeperm`)

name	default	description
`niter`	1	number of iterations of node permutation algorithm per generation
`verbose`	0	level of logging from individual evolution progress 0 – no logging 1 – some logging 2 – plus timing info from `calc_injection_delta` 3 – plus logging from LP solver inside `calc_injection_delta`
`nox`	1	assumption used when solving LP in `calc_injection_delta` 0 – use full sensitivities 1 – use decoupled power flow assumption
`usedv`	0	consider voltage deviation constraints in LP in `calc_injection_delta` (see Section 5.5).
`scale_s`	1	scaling factor for branch rating: $f_i^{\text{new}} = \texttt{scale\_s} \cdot f_i$
`scale_s_factor`	0.95	factor to reduce `scale_s` over iterations: $\texttt{scale\_s}^{\text{new}} = \texttt{scale\_s\_factor} \cdot \texttt{scale\_s}$.

Table 6-9: Reactive Planning Options (`vm.shunts`)

name	default	description
`verbose`	0	level of logging from individual evolution progress 0 – no logging 1 – some logging
`tmag`	0.1	minimum size of shunt in MVAr
`shift_in`	0.015	amount voltage limits are shifted in to try to avoid binding voltage constraints after soft limits are removed
`shunt_max`	500	maximum shunt size in MVAr

Appendix A Release History

The full release history can be found in `CHANGES.md` or online at `https://github.com/MATPOWER/mx-syngrid/blob/master/CHANGES.md`.

A.1 Version 1.0 – released December 18, 2018

The SynGrid 1.0 User's Manual is available online.[14]

- Initial release.

[14]`http://www.pserc.cornell.edu/matpower/docs/SynGrid-manual-1.0.pdf`

References

[1] Z. Wang, R. J. Thomas and A. Scaglione, "Generating Random Topology Power Grids," *Proceedings of the 41st Annual Hawaii International Conference on System Sciences (HICSS 2008)*, Waikoloa, HI, 2008, pp. 1–9. https://doi.org/10.1109/HICSS.2008.182 1.1

[2] Z. Wang, A. Scaglione and R. J. Thomas, "Generating Statistically Correct Random Topologies for Testing Smart Grid Communication and Control Networks," *Smart Grid, IEEE Transactions on*, vol. 1, no. 1, pp. 28–39, June 2010. https://doi.org/10.1109/TSG.2010.2044814 1.1, 1.3, 4.1

[3] Z. Wang and R. J. Thomas, "On Bus Type Assignments in Random Topology Power Grid Models," *2015 48th Hawaii International Conference on System Sciences*, Kauai, HI, 2015, pp. 2671–2679. https://doi.org/10.1109/HICSS.2015.322 1.1

[4] Z. Wang, S. H. Elyas, "On the Scaling Property of Power Grids," *Proceedings of the 50st Annual Hawaii International Conference on System Sciences (HICSS 2017)*, Waikoloa, HI, 2017, pp. 3148–3155. https://doi.org/10.24251/HICSS.2017.381 1.1

[5] S. H. Elyas, Z. Wang, "Improved Synthetic Power Grid Modeling with Correlated Bus Type Assignments," *Power Systems, IEEE Transactions on*, vol. 32, no. 5, pp. 3391–3402, Sept. 2017. https://doi.org/10.1109/TPWRS.2016.2634318 1.1, 1.3, 4.2

[6] E. Schweitzer, A. Scaglione, "A Mathematical Programing Solution for Automatic Generation of Synthetic Power Flow Cases," *Power Systems, IEEE Transactions on*, 2018. https://doi.org/10.1109/TPWRS.2018.2863266 1.1, 1.3

[7] S. H. Elyas, Z. Wang, R. J. Thomas,"On the Statistical Settings of Generation Capacities and Dispatch in a Synthetic Grid Modeling," *10th Bulk Power Systems Dynamics and Control Symposium (IREP'2017)*, Espinho, Portugal, Aug 27–Sep 1, 2017. 1.1

[8] S. H. Elyas, Z. Wang, R. J. Thomas, "On the Statistical Settings of Generation and Load in a Synthetic Grid Modeling," *arXiv preprint arXiv:1706.09294*, 2017. 4.3

[9] H. Sadeghian, S. H. Elyas, Z. Wang, "A Novel Algorithm for Statistical Assignment of Transmission Capacities in Synthetic Grid Modeling," *2018 IEEE PES General Meeting*, Portland, OR, Aug 5–9, 2018. 1.1, 4.3, 4.4

[10] Z. Wang, M. H. Athari, S. H. Elyas, "Statistically Analyzing Power System Network," *2018 IEEE PES General Meeting*, Portland, OR, Aug 5–9, 2018. 1.1

[11] John W. Eaton, David Bateman, Søren Hauberg, Rik Wehbring (2015). *GNU Octave version 4.0.0 manual: a high-level interactive language for numerical computations.* Available: `http://www.gnu.org/software/octave/doc/interpreter/`. 1, 5

[12] The BSD 3-Clause License. [Online]. Available: `http://opensource.org/licenses/BSD-3-Clause`. 1.2

[13] Z. Wang, H. Sadeghian, S. H. Elyas, and R. D. Zimmerman, "SynGrid User's Manual," 2018. [Online]. Available: `http://www.pserc.cornell.edu/matpower/docs/SynGrid-manual-1.0.pdf` 2.4

[14] R. D. Zimmerman and C. Murillo-Sánchez. Matpower User's Manual, [Online]. Available: `http://www.pserc.cornell.edu/matpower/`

[15] R. D. Zimmerman, C. E. Murillo-Sánchez, and R. J. Thomas, "Matpower: Steady-State Operations, Planning and Analysis Tools for Power Systems Research and Education," *Power Systems, IEEE Transactions on*, vol. 26, no. 1, pp. 12–19, Feb. 2011. `http://dx.doi.org/10.1109/TPWRS.2010.2051168`

[16] A. B. Birchfield, T. Xu, and T. J. Overbye, "Power Flow Convergence and Reactive Power Planning in the Creation of Large Synthetic Grids," *Power Systems, IEEE Transactions on*, vol. 33, no. 6, pp. 6667–6674, Nov. 2018. `http://dx.doi.org/10.1109/TPWRS.2018.2813525` 5.6

[17] T. Xu, A. B. Birchfield, K. M. Gegner, K. S. Shetye, and T. J. Overbye, "Application of large-scale synthetic power system models for energy economic studies," *In Proceedings of the 50th Hawaii International Conference on System Sciences*. 2017.

[18] E. Schweitzer, A. Scaglione, R. Thomas, and T. Overbye, "Analysis of the Coupling Between Power System Topology and Operating Condition for Synthetic Test Case Validation," *in Proceedings of the 2016 CIGRE Grid of the Future Symposium*. 2016.

5.2