Example Distribution Network Test Case Documentation

Dakota Thompson
Dakota.J.Thompson.TH@Dartmouth.edu
Thayer School of Engineering, Dartmouth College, 14Engineering Drive, Hanover, NH, 03755

Introduction

The code described below can be used to recreate the test case, Hetero-functional graph theory (HFGT) compliant, XML input file that was used in the Innovative Smart Grid Technologies (ISGT) 2020 conference paper "A Hetero-functional Graph Analysis of Electric Power System Structural Resilience"[1]. This test case was based off the IEEE 123-Bus Feeder test case [2] . After closing and converting switches to distribution lines to represent a radial distribution system the resulting test case modeled is depicted in Fig. 1. This test case is describing with two csv files, one for the nodes and one for the lines. These csv files are then read in and processed to create the HFGT compliant XML file using the code described below.

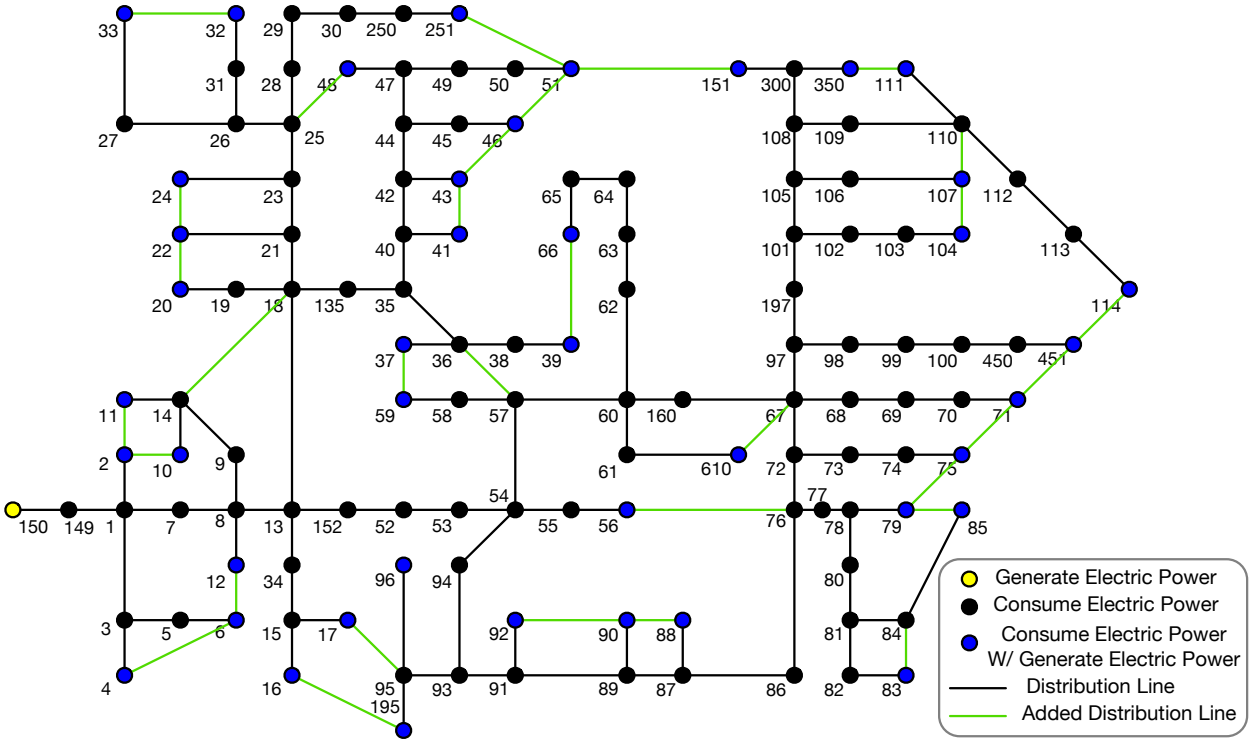

Figure 1. This figure depicts the test case based off of the IEEE 123-Bus feeder test case with added distributed generation capabilities and meshed distribution lines.

Code Documentation

The python script distributed_grid.py is used to read in the csv files, process the data into a grid object, then export a HFGT compliant XML of the test case. This script was developed in python version 3.7 and can be run in terminal by calling:

$: python distributed_grid.py outputFileName.xml

1) Distributed_grid.py: This script describes the DistributedGrid class. The required python packages are: pandas, lxml, collections.OrderedDict, and sys. It also requires the scripts ElecNode.py, and ElecLine.py to be available in the working environment. The DistributedGrid class has the following methods described below.

1.1) read_in_csv(self, files, rootDir): This method is used to read in the csv files describing the node locations and the distribution lines. The input "files" is a list of strings giving the csv file names. The input "rootDir" is a string giving the relative path to the input files folder. After calling this method the grid object is populated with all of the test case nodes and edges.

1.2) add_lines(self, file, rootDir): This method is used to add meshed distribution lines to the grid object. The input "file" is a string giving the name of a csv file containing the origins and destinations of the meshed distribution lines to be added to the grid. The input "rootDir" is a string giving the relative path to the input files folder.

1.3) add_DG(self, file, rootDir): This method is used to add distributed generation functionality to consumption nodes in the grid object. The input "file" is a string giving the name of a csv file containing the names of all the nodes that are to be given distributed generation capabilities. The input "rootDir" is a string giving the relative path to the input files folder.

1.4) write_hfgt_xml(self, filename): This method is used to write the resulting HFGT compliant XML. The input "filename" is a string giving the name of the output XML to be written. Looping over each node and line each resource is added to the XML root with their allocated functionality.

2) ElecNode.py: This script describes the ElecNode class. The required packages are lxml, and collections.OrderedDict. This object is used to represent nodes in the distributed grid. It has the following methods.

2.1) get_coordinate(self): This method returns the nodes gps coordinates.

2.2) add_xml_child_lfes(self, parent): This method adds the node object and its allocated functionality to the input XML root. The input parent represents the root of the xml being written.

3) ElecLine.py: This script describes the ElecLine class. The required packages are lxml, and collections.OrderedDict. This object is used to represent lines in the distributed grid. It has the following method.

3.1) add_xml_child_lfes(self, parent): This method adds the line object and its allocated functionality to the input XML root. The input parent represents the root of the xml being written.

Data Documentation

The four csv files included with this code are used to create the test case xml and its variants with added distributed generation and meshed distribution lines.

1) node_data.csv: This file describes each node name, gps coordinates, and the node type. It is used when constructing the test case adapted from the IEEE 123-Bus Feeder test case [2].

2) line_data.csv: This file describes the node origin, node destination, and length of each line in the test case. It is used when constructing the test case adapted from the IEEE 123-Bus Feeder test case [2].

3) addedDG.csv: This file gives the name of each node that is to be allocated distributed generation capabilities. The most radial nodes with only one connection were designated to receive distributed generation. They are identified in Fig. 1.

4) addedLines.cav: This file gives the names of the node origins and node destinations of the meshed distribution lines to be added to the grid. Meshed distribution lines were designated so as to further connect in nodes with previously only one connection to the grid. They are identified in Fig. 1.

References

[1] Thompson, Dakota J., Wester CH Schoonenberg, and Amro M. Farid. "A hetero-functional graph analysis of electric power system structural resilience." *2020 IEEE Power & Energy Society Innovative Smart Grid Technologies Conference (ISGT)*. IEEE, 2020.

[2] Kersting, William H. "Radial distribution test feeders." *IEEE Transactions on Power Systems* 6.3 (1991): 975-985.