American Electric Power System 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 IEEE Access journal paper "A Hetero-functional Graph Resilience Analysis of the Future American Electric Powe System"[1]. This test case was created using the Platts Map Data Pro Datasets 2016 electric grid layers [2]. After processing and cleaning the GIS layers into a HFGT compliant XML, the file can be converted into a HFG and resilience analysis can be run. The text below describes the python scripts used to produce the XML.

Code Documentation

The python script kml_to_hfgt_xml.py is used to convert the GIS layers kml files to a HFGT compliant XML. The kml_to_hfgt_xml.py script calls upon an ElecGrid object to store the data in an object oriented design before producing the XML. The code was developed using python version 3.7.

1) kml_to_hfgt_xml.py: This script is used to create the ElectricGrid object then write the output xml. The required packages are sys and the ElecGrid class. It is call via the command:

$: python kml_to_hfgt_xml.py outputFileName.xml

Once called the script begins by creating the electric grid object. Once instantiated the object is populated by the kml files. The populated grid then removes any isolated nodes, transmission lines, and subcomponents from the grid object. Finally, the remaining resources are used to write the output HFGT compliant XML

2) ElecGrid.py: This script defines the ElecGrid class that is used to store the US electric grid data. It's required packages are numpy, scipy.io, lxml, collections.OrderedDict, shapely.geometry.LineString, snap, and access to the scripts geometry/InstanceLayer.py, elec/ElecNode.py, elec/ElecLine.py, elec/ElecSubstation.py, elec/ElecGenUnit.py, and elec/ElecPoerPlant.py. This class contains the methods required to instanciate the electric grid, clean the electric grid data, then write out the xml. Its methods are described below.

2.1) read_in_kml(self, files, rootDir): This method is used to read in the kml data and populate the electric grid. The input "files" is a list of strings giving the kml file names of the layers to be incorporated int eh electric grid. The input "rootDir" is a string providing the relative path to the kml files. This method calls the functions self.read_powerPlants(), self.read_genUnits(), self.read_substations(), and self.read_transmissionLines() to populate the grid object with each of the input kml files.

2.2) read_powerPlants(self, filename, buffer_map):This method is used to read in the GIS power plant layer's kml file to populate the electric grid. The input "filename" is a string

representing the kml file's relative location and name. The input "buffer_map" is a set of existing coordinates already populating the ElecGrid object. This method loads in and stores existing and functioning powerplants into the ElecGrid object as ElecPowerPlant objects. It returns the updated buffer_map set as well as the number of instantiated powerplants and storage units.

2.3) read_genUnits(self, filename, buffer_map, plant_count): This method is used to read in the GIS Generation Unit layer's kml file to populate the electric grid. The input "filename" is a string representing the kml file's relative location and name. The input "buffer_map" is a set of existing coordinates already populating the ElecGrid object. This method loads in and stores existing and functioning generation units into the ElecGrid object as ElecGenUnit objects. It returns the updated buffer_map set.

2.4) read_substations(self, filename, buffer_map): This method is used to read in the GIS Substation layer's kml file to populate the electric grid. The input "filename" is a string representing the kml file's relative location and name. The input "buffer_map" is a set of existing coordinates already populating the ElecGrid object. This method loads in and stores existing and functioning substations into the ElecGrid object as ElecSubstation objects. It returns the updated buffer_map set.

2.5) read_transmissionLines(self, filename, buffer_map): This method is used to read in the GIS Transmission Line layer's kml file to populate the electric grid. The input "filename" is a string representing the kml file's relative location and name. The input "buffer_map" is a set of existing coordinates already populating the ElecGrid object. This method loads in and stores existing and functioning transmission lines into the ElecGrid object as ElecLine objects.

2.6) remove_isolated(self): This method identifies all the isolated nodes in the ElecGrid object and removes them.

2.7) remove_subComps(self): This method uses the Stanford University Snap.py toolbox to identify all of the subcomponents of the ElecGrid object [3]. After identifying the subcomponents of the grid all but the main component are removed.

2.8) write_hfgt_xml(self, filename): This method is used to write the HFGT compliant xml. The input "filename" is a string representing the name of the resulting xml file. After forming the element tree this method calls the power_node.add_xml_child_hfgt() to populate the nodes of the xml and then iterates over each transmission line adding them to the tree as well. Finally, the abstraction is written and the xml file is saved.

3) InstanceLayer.py: This script is used to load in the kml file into a format that can then be easily interfaced with. An InstanceLAyer object is first instantiated as an empy object then populated by calling it's build_from_QGIS() method on the input kml filename. An example usage is shown below:

```
Layer = InstanceLayer()
Layer.build_from_QGIS(ET.parse(kmlFileName))
```

4) ElecNode.py: This script is used to define the ElecNode object to hold all of the nodes in the electric grid. This object has attributes to store a list of GenC units, GenS units, LoadC units, LoadS units, StorageC units, StorageS units, Substations, Independent Buffers, and node attributes. As the electric grid becomes populated nodes are added to their respective lists in the ElecNode object. The ElecNode object has a method "add_xml_child_hfgt(self, parent)" which is used to iterate over every node and add the resource and their respective functions to the parent xml file.

5) ElecLine.py: This script is used to define the ElecLine objectused to define transmission lines in the electric grid. When instantiating, this object takes an input of the instantiated line's attributes and its coordinates. The ElecLine object has a method "add_xml_child_LFES(self, parent)" which is used to add the line and their respective functions to the parent xml file.

6) ElecSubstation.py: This script is used to define a substation object to hold the data associated with a substation instance.

7) ElecSubstation.py: This script is used to define a Generation unit object to hold the data associated with a generation unit instance.

8) ElecSubstation.py: This script is used to define a power plant object to hold the data associated with a power plant instance.

References

[1] Thompson, Dakota J., Wester CH Schoonenberg, and Amro M. Farid. "A Hetero-Functional Graph Resilience Analysis of the Future American Electric Power System." *IEEE Access* 9 (2021): 68837-68848.

[2] Platts, "Platts energy map data pro," S&P Global Platts, Tech. Rep., 2017.[Online]. Available: http://www.platts.com/products/map-data-pro

[3] J. Leskovec and R. Sosivc, "Snap: A general-purpose network analysisand graph-mining library,"ACM Transactions on Intelligent Systems andTechnology (TIST), vol. 8, no. 1, p. 1, 2016.