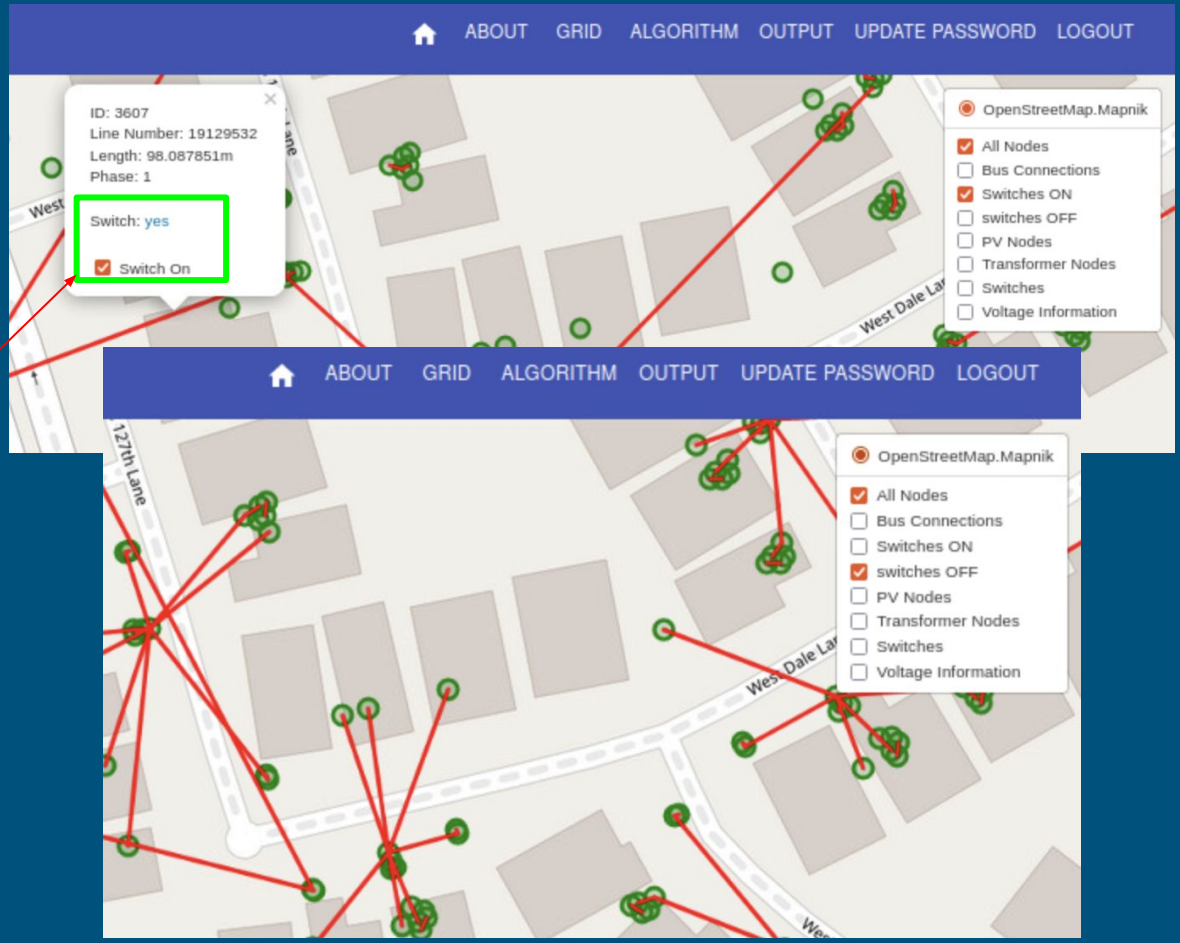
AI-VVO sdmay22-36 Weekly Update #10

4/7/2022 - 4/14/2022

Front-end (This Week)

- Changed colors of nodes based off of their type
- Displayed node voltage information onto popups
- Updated Switch lines to have checkbox selector for ON/OFF
- Added 2 new layers to display switched on lines and switched off lines
- Added documentation into updated front-end files

- New checkbox for clarity on whether switch is on or off
- 2 new layers for Switch ON and Switch OFF which show only those corresponding lines



Front-end (Next Week)

- Continue adding documentation to front-end files
- Display power and current information onto bus line popups
- Work on updating algorithm page
- Look into loading data faster

Back-end (This Week)

- Fix bug that was causing switch update to not work properly
- Started to create unit tests for the api endpoints

Back-end (Next Week)

- Create more unit tests to try and get full code coverage on the backend
- Help make sure that the connect between the frontend and backend works properly

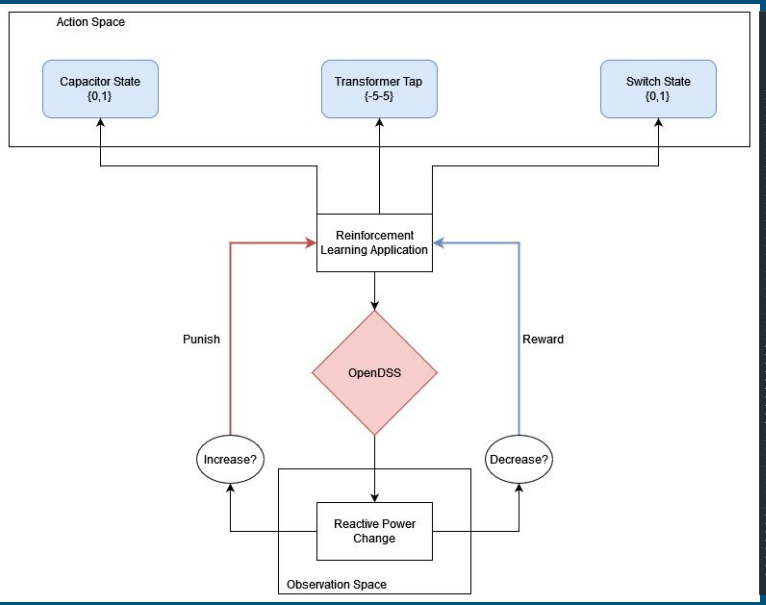
Dockerfile (This Week)

- Implement changes to switch to remove multi-stage builds

Dockerfile (Next Week)

- Clean up the Dockerfiles/add more in-text documentation.

Machine Learning (This Week)




```python
#Group: sdmay22-36
#Title: ReactivePowersPull (Pulls Reactive Powers from the OpenDSS Instance)
#Authors: Evan Dinnon, Will Dulaney
#Date: 03/06/2022 (updated 4/09/2022)


import os.path

import py_dss_interface


current_path = os.path.abspath(os.getcwd())

# Specify the OpenDSS file path. Different per machine
open_dss_path = "D:/Programs/OpenDSS"

# Initialize the python OpenDSS interface
dss = py_dss_interface.DSSDLL(open_dss_path)

# Select DSS Model from Current Working Directory
dss_file = current_path + "\\CLW13_March15_v6\\Master_v6_March15.dss"


# Use Text command to compile the DSS script file
dss.text("compile {}".format(dss_file))

# Using .text we can control every aspect of OpenDss
# There are individual methods we can call as well

# Here is the standalone method for solving the dss circuit
dss.solution_solve()

# This opens a text file showing the voltages
dss.text("show voltages")


import os.path

import py_dss_interface

def PullPowers():
    path = os.path.abspath(os.getcwd())
    current_path = os.path.abspath(os.getcwd())

    # Specify the OpenDSS file path
    open_dss_path = "D:/Programs/OpenDSS"

    # Initialize the python OpenDSS interface
    dss = py_dss_interface.DSSDLL(open_dss_path)

    # Select DSS Model from Current Working Directory
    dss_file = path + "\\CLW13_March15_v6\\Master_v6_March15.dss"


    # Use Text command to compile the DSS script file
    dss.text("compile {}".format(dss_file))

    # Solving method
    dss.solution_solve()

    # This exports a text file showing the powers. Reactive powers can then be parsed to determine the loss function
    dss.text("export powers")

    #Return the sum of all reactive powers to give loss function
    return powers.sum()
```

- Refactored Git workspace to have more meaningful filenames for the ML scripts
- Reactive powers pulled by updated OpenDSSControl.py script.
- Main learning environment created using the “Gym” Library
- Reward observed as the previous total reactive power subtracted from the updated reactive power

Machine Learning (Next Week)

- Develop the Machine learning agent that will utilize the environment created
- Iron out any bugs found in the environment from testing with the agent
- Make algorithm’s startpoint contingent upon input from the pushing of a front-end GUI button