SANTA CLARA UNIVERSITY

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

Date: June 7, 2021

I HEREBY RECOMMEND THAT THE THESIS PREPARED UNDER MY SUPERVISION BY

Christopher Woo

ENTITLED

Smart Grid Security Simulator

BE ACCEPTED IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE OF

BACHELOR OF SCIENCE IN COMPUTER SCIENCE AND ENGINEERING

Xiang Li

Thesis Advisor

Li Xiang

Thesis Advisor

Nam Ling

Department Chair

Nam Ling

Nam Ling (Jun 8, 2021 08:32 PDT)

Department Chair

Smart Grid Security Simulator

by

Christopher Woo

Submitted in partial fulfillment of the requirements
for the degree of
Bachelor of Science in Computer Science and Engineering
School of Engineering
Santa Clara University

Santa Clara, California
June 7, 2021

Smart Grid Security Simulator

Christopher Woo

Department of Computer Science And Engineering
Santa Clara University
June 7, 2021

ABSTRACT

Visualizing the cascading failure of a network can be challenging due to the complexity of smart grid functionalities. The web application, Smart Grid Security Simulator (SGSS), is designed to make it easier for people, who intend to learn about smart grid security, to better visualize the failure of lines in a network due to various attacks. By utilizing SGSS, users will be able to learn what types of attacks existing networks are prone to and how different modifications to existing networks can bolster their security against each type of attack.

Acknowledgments

I would like to thank my advisor, Prof. Xiang Li, who has given me terrific guidance throughout the course of this project.

Table of Contents

List of Figures

Chapter 1

Introduction

1.1 Problem

Natural disasters and targeted attacks can cause power lines to fail. The failure of a line in a power grid can result in more stress for other lines and nodes in the grid. Due to the increase of current through various lines and the limited capacities of the lines, multiple lines can fail due to the initial line failure. This process is known as cascading failure and may continue throughout the network until the network is stabilized.

In the worst case, natural disasters and targeted attacks on power grids can result in major blackouts if the emergency is not handled properly. A power grid blackout causes major social, economic, and political unrest. More specifically, it can cause disruptions in medical systems, road traffic, payment transactions, security systems, and the internet. This affects almost everyone in their modern daily lives.

Many power grid networks across the world have suffered from frequent occurrences of blackouts in the past decades. In USA alone, in the total number of blackouts recorded from 2008 to 2015, 2169 to 3236 power outages were recorded annually with a minimum of 13 million people being affected or left without power completely [1]. Generally, these blackouts were caused by natural disasters, old power systems, and operation failures of protection systems [1].

One of the most notable power grid failures in America over the past decades was the 2003 Northeast Blackout. After some power lines had encountered overgrown trees, the lines softened under the heat of high current going through them, finally switching off. Due to an alarm system failure, the Ohio-based utility company was not able to react in time. After tripping a cascading failure effect through southeastern Canada and 8 northeastern states, around 50 million people lost power for up to 2 days, resulting in 11 deaths and costing around 6 billion dollars [1,2,3].

It is crucial for power grid blackouts to be avoided completely, but how is that possible?

1.2 Solution

To start, it is important to better understand what power lines will fail based on whether a natural disaster or a targeted attack hit the power grid. The Smart Grid Security Simulator (or SGSS), helps users visualize what lines will fail based on where the attack is initiated. It then plays an animation, showing which lines will continue to fail on each time-step. The line failure in SGSS is calculated by checking if the current going through each line is greater than the calculated capacity of each line. If the current is greater, the line fails in that time-step. This feature helps users find lines with more importance to the integrity of the power grid, allowing them to figure out which lines need to be expanded.

The SGSS also allows users to draw new lines and nodes as well as modify certain parameters of individual lines and nodes in order to help users better understand how a network can be improved upon. This can be done by using the provided user interface of the SGSS simply by inputting the voltages or capacities that the user would like to provide. Through this tool, users will be able to learn what modifications can be made to a network in order for it to be more resilient to various attacks. The information gathered in this project can also be used to design various strategies that will strengthen existing networks against natural disasters and targeted attacks on individual lines in a network.

1.3 Existing Studies

Currently, there exists many studies [1,4,5] that helps readers to better understand how different networks respond to various attacks. These studies show the number of lines failed as well as providing information about the importance of each line. Existing studies generally require much more in-depth knowledge as to how smart grids function as well as how networks react to various attacks [4,5]. Due to this, people learning about smart grids and cascading failures will find it difficult to truly test various changes to a network. To combat that problem, the Smart Grid Security Simulator is designed to provide a more simplified user interface that allows modification of existing networks which provides users with a more learning-oriented view on cascading failure results due to various attacks.

1.4 Cascading Failure

To give a brief description on cascading failures in a power grid, essentially a failure in one or more interconnected parts of a grid can trigger following failures in other parts of a power grid. The failure of the initial line in a power grid results in the re-balancing of power generation and load of buses in the network. This

re-balancing can cause the power flow of other lines in a power grid to be greater than the line's capacity [4]. When this happens, the lines may fail, resulting in further lines to experience the same effect.

Chapter 2

Project Development

2.1 Development Timeline

Fall Quarter - *Finish research to fully understand the topic*	
By end of week 10	Research/read articles on smart grid functionality as well as smart grid cascading failure behavior in order to better understand how a smart grid works and how it will react to natural disasters or directed attacks.
Winter Break - *Begin development of project*	
During winter break	Begin coding.
Winter Quarter - *Continue development of project, finishing the foundation of the development process*	
Week 1	Complete the visualization functionality for the web application. The application should allow a user to select a network from the user interface and load it from the dataset in the backend.
Week 4	Design and develop a way to play an animation showing what lines fail at each time step given that an attack on the smart grid occurs. This step should allow a user to visualize the cascading failure behavior.
Week 7	Implement different cascading failure models in the backend of the program. This should determine which lines fail first as well as how lines continue to fail after the initial attack has occured.
Week 8	Implement a way in the frontend of the program in order to allow users to select which cascading failure model to use. This should also provide users with the ability to decide the initial location of the attack.
Week 9/10	Add support to create or edit the network using the UI. This step should allow a user to add nodes or lines as well as allow users to change network parameters such as increasing or decreasing line capacities.
Spring Quarter - *Debugging/fixing remaining problems. Additional Features.*	
By week 5	Finish up work on the foundation of the program. Solve any remaining problems with the program from the development process during winter quarter. If there is time remaining, add function to the program that allows users to visualize various strategies that may further protect a network against various attacks (such as choosing 10 lines with the highest importance and doubling their capacity).

Figure 2.1: Development Timeline

2.2 Risk Identification

There were a few potential risks that may have hindered the development of the project. These risks include the failure to properly implement the ability to calculate the cascading failure simulation. This specific risk may occur due to a misconception of how a smart grid will react to different scenarios. In this case, the program would be providing users with a false idea of how a smart grid works, this would be considered a failure but would be hard to detect. This could occur due to a lack of information or knowledge in how each attack impacts a network.

Another risk that needed to be considered was the failure to implement the visualization of cascading failure. Although this was less of a concern due to the use of a prototype provided by Dr. Xiang Li, it is important that the visualization of the lines failing corresponded to the correct time step. This risk is notable due to the possible problem of how data is transferred between JavaScript and Python. A line that is calculated to fail under the implemented Python functions should correspond to the same line that is visualized in the front end by the JavaScript models.

2.3 Risk Mitigation

Both of the risks that were identified in the previous section were ones that I had handle throughout the development process. In order to properly implement the cascading failure model for the SGSS, I did extensive research on the various models that have been used in order to analyze smart grids in the past. By doing so, I was able to select models that could be accurately simulated in the SGSS. The model that I ended up using for cascading failure is called Generation Ramping [4]. This model will be described more in detail in a later section.

When it came to the risk of the implementation for the visualization of cascading failure in the front end, I was able to work around this risk by trial and error. Initially, the lines that were visualized to be failing were not the ones that had been calculated by the back-end. This was due to a fourth type of node that I had not initially accounted for. The three main types of nodes that are in each of the networks provided in pandapower are a bus, a generator connected to a bus, and a load connected to a bus. However, some networks, such as the 30-bus network that I had used in testing, contained a bus that was connected to both a generator and a load. Although this took a lot of debugging to figure out, the problem ended up being rather simple to solve. I worked around this by adding another checker in order to figure out if a bus was connected to both a load and generator, and reformatted the JSON being sent to JavaScript to allow for this type of node to exist. Finally,

I added a new icon to represent this fourth type of node.

2.4 List of Requirements

Necessary
Ability to select a network from the UI and load it from dataset in backend (functional)
Allow user to visualized cascading failure behavior (functional)
Allow user to choose from different cascading failure models in frontend (functional)
Process how each cascading failure model will function in backend, translate to visualization in frontend (functional)
Support the creation or editing of existing networks in the UI, giving users the ability to add nodes and lines as well as change network parameters (functional)
Recommended
Provide users with a way to visualize different strategies to protect the network against the attack
Calculate and provide users with the best strategy available given each scenario (based on location and type of attack)
Allow users to upload their own networks

Figure 2.2: Table of Requirements

Based on the project design, the figure above, figure 2.2, is a list of requirements that were measured and observed during the development of the project. The list of requirements is separated into those that are necessary and those that can be added but are not critical to the design of the project. The main requirements for the project are considered functional requirements, meaning that they were able to be observed throughout the development process.

As for the non-functional requirements for the project, the program should be designed to be user-friendly as well as engaging. Since SGSS will mostly be used as a tool to aid in teaching or learning how smart grids respond to different attack cases, it is important that the animations used in the program keep users engaged while also being intuitive. The UI should be user-friendly in that the user should have no issues navigating through the different functionalities of the program. More specifically, the user, at the very least, should be able to easily select a network from the list provided, choose an attack scenario, as well as choose the location that they want the initial attack to take place. This should all be done without the user struggling to find any of these individual components without having much knowledge on smart grids. However, it can be assumed

that some sort of understanding of networks and smart grids are needed when it comes to creating and editing a network.

2.5 Meeting the Requirements

Currently, the web application allows users to select networks and attacks from the front end and calculate the attacks in the backend. The user is also able to modify the power output or requirements of all nodes and the capacity of all lines in the network. These nodes consist of generators, loads, and buses. SGSS also provides users with the ability to select whether the cascading failure model will be calculated using AC power flows or DC power flows, where the following line failures are then calculated in the backend and visualized in the front end.

The SGSS is not yet capable of providing users with strategies to protect the network against further attacks. However, I plan on implementing this feature in the next stage of development for the SGSS.

2.6 Benefits

In its planned final form of implementation, SGSS can provide many benefits. Firstly, it can provide users learning about cascading failure with a better understanding of smart grids and their reaction to various attacks. When network improvement suggestions are implemented, SGSS can also help power companies better manage power grids by providing information on which lines or nodes should be improved upon to strengthen the security of the network. This can result in economic benefits as cascading failures in power grids will happen less frequently. The strengthening of power grids could also prevent events such as the Northeast Blackout from being as severe. Potentially saving lives and improving safety for residents who rely on the given power grid.

Chapter 3

Concept

3.1 Use Cases

The Smart Grid Security Simulator web application is designed to help those who are learning about smart grids to better visualize how a smart grid functions and how cascading failure works based on various attack cases. Additionally, it is intended to help users locate potential flaws in existing networks and provide them with strategies to improve the security of the network against each type of attack. The use case diagram is shown in figure 3.1.

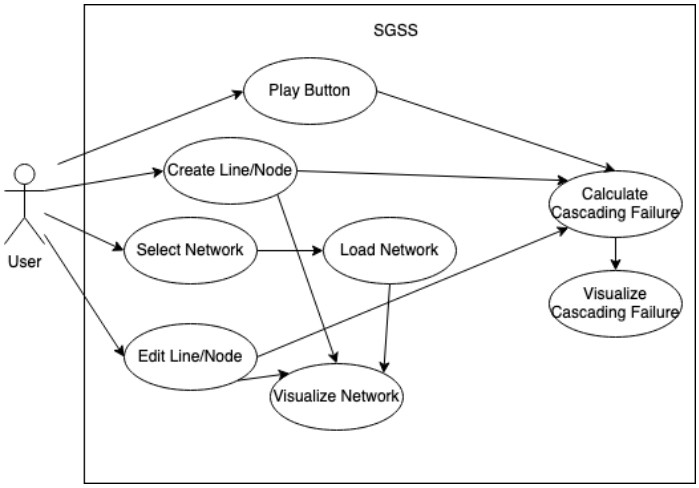

Figure 3.1: Use Case Diagram

Use Case 1 - Learning about smart grid security

Actor: Student/person learning about smart grid functionality

Basic Flow: Student launches web application and selects a network that they would like to visualize. They choose which type of attack that they would like to view the cascading failure for and selects a specific

area/group of lines for the attack to initialize at. SGSS shows which lines fail at each time-step.

Use Case 2 - Teaching smart grid security

Actor: Teacher/person teaching about smart grid functionality

Basic Flow: Teacher launches web application and selects a network that they would like to visualize. Similar to how students would use the application, but in addition, they can show students how different modifications on the network can impact a network positively or negatively. Additionally, they can modify existing networks or networks that they created and view the same attack side by side with the unmodified version of the network. This will help them show the difference that modifications to certain lines in a network can make compared to the approximate cost that it will take to make that modification. By doing so, this feature can be used to find possible improvements on existing networks or networks that have been created by the user.

3.2 Components

The Smart Grid Security Simulator application consists of a front-end and a back-end.

The main components of the front-end include the prompts that will ask the user to select a network or edit a network, as well as selecting an attack case and the location of the attack. The attack size can vary by number of lines and the attack cases consist of a natural disaster attack and a targeted attack. The natural disaster attack selection selects all lines connected to the node that a user selects. The targeted attack selection allows a user to select individual lines to fail. Another important component of the front-end is the visualization of a network as well as the visualization of the cascading failure effect through each time-step.

The main components of the back-end consist of the communication between what the user inputs in the front-end and the functions in the back-end. This will allow data to be pulled from the pandapower package in order to load the network that is specified by the user. Additionally, the back-end consists of a cascading failure component that determines which lines will fail after the initial attack. This then communicates with the front-end to allow the visualization of which lines proceed to go out at each time-step. The SGSS allows a user to interact with the provided UI in order to create and edit networks to determine strengths and weaknesses in their modified networks.

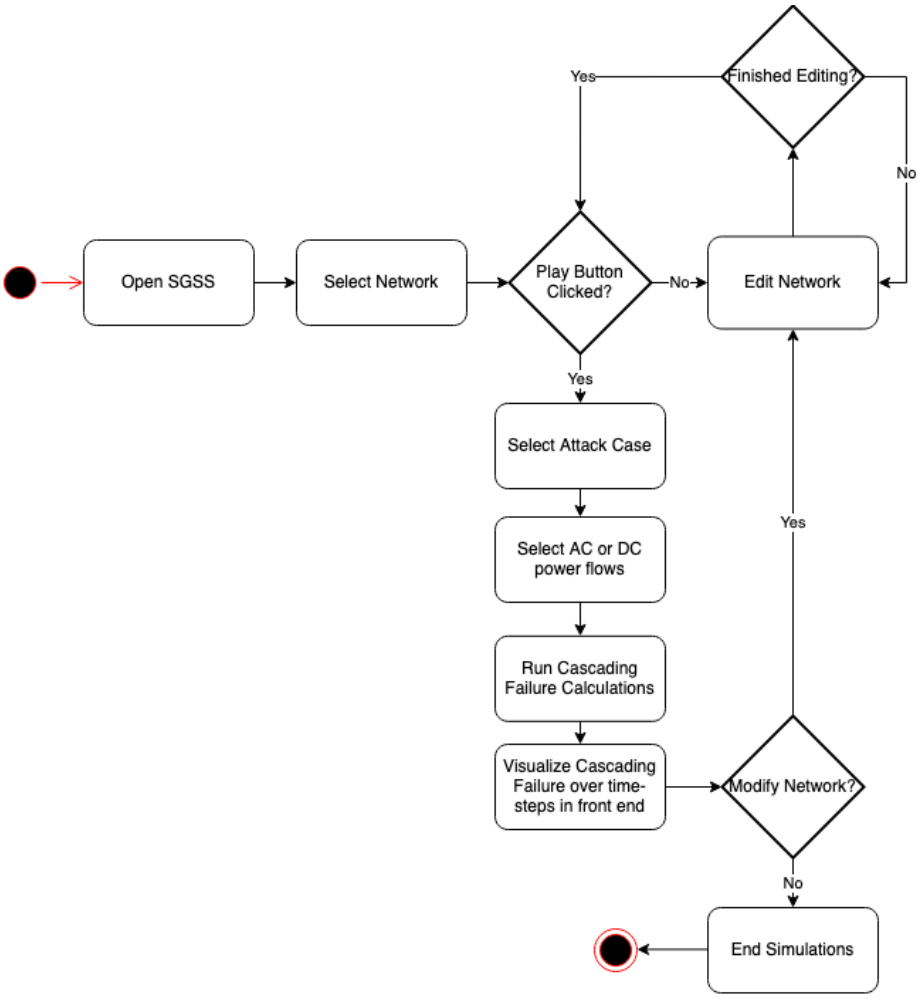

Figure 3.2: Activity Diagram

3.3 Activity Diagram

The Activity Diagram in Figure 3.2 shows the processes that a user can go through in order to select, modify, and visualize a cascading failure effects from the given power flow models and attack cases.

3.3.1 Selecting a Network

The user begins by opening SGSS and selecting a network for which they would like to view. The network can be changed by the user after starting. A change in the network will essentially act as a complete refreshment of the application and all data that has been inputted. Once a network is selected, the user can click on the play button in order to start the calculation process. The user can also choose to edit a network before selecting to play the simulation. Since the initial networks provided by pandapower are all stable, clicking play with a

default network and no attack cases will result in no lines failing.

3.3.2 Editing a Network

A user may add nodes or lines to the network that they selected. The user should specify the voltages for the load and generators if they choose to add one, and input a capacity for every line that is added. A user may also use the same input boxes in order to modify existing nodes or lines of the network. Once the user is done editing the network to their liking, they can simply click the play button to proceed with the cascading failure simulation.

3.3.3 Cascading Failure Simulation

Once the play button is clicked, the user can choose an attack case. The user may choose between natural disaster attacks and targeted attacks. The natural disaster attack option will select all lines that connect to the node that a user selects and the targeted attack option allows the user to select individual lines. The user may select as many lines as they would like for the initial attack, as well as use both natural disaster attacks as well as targeted attacks in the same attack case. The network will have no attacks by default, so this must be selected by the user.

The user can also select whether the cascading failure calculations use AC or DC power flows. The default for this option choice is the AC power flow. The power flow selection is used in the calculations by running a power flow through the network at each time-step in order to determine whether or not the current flow in a line exceeds the capacity of the line.

Once the attacks and power flow are selected, the formatted JSON file is sent to the back end for calculations. The Python script will run calculations on each time step as well as set up the initial time steps for visualization. Once these calculations are completed, the Python script re-formats the new data into a JSON file to send to the front-end for visualization.

Finally, the front-end visualizes all the line failures for each time-step in the network and plays the animation from time-steps 1 through 10 for the user. Once the simulation is complete, the user can choose to modify the network and run the process again by clicking the play button. If the user does not do this, then the user is assumed to be done using the application. The user may also click the play button again without editing the network to simply replay the simulation that has already run

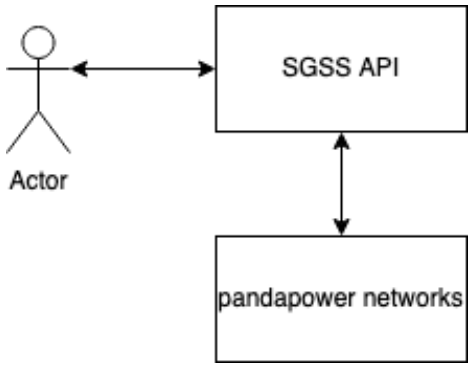

Figure 3.3: Simple Architectural Diagram

3.4 Simple Architectural Diagram

The SGSS program works with basic steps, as user authentication is not required since no information is being protected. The program will only have to determine that the actor is communicating properly with the server. Figure 3.2 shows how these will communicate with each other. The actor will be communicating directly with the SGSS API. The SGSS API will grab data from the Python package pandapower in order to retrieve networks and load it to be visualized.

3.5 Design Rationale

Before beginning the development stage of the project, I had done a lot of research on smart grid functionalities as well as cascading failure models based on different attack scenarios in order to have a better understanding of how each work. This is important in order to most accurately simulate these events in the program.

I had chosen to work with a more simplistic UI as well as using color-coding for each of the inputs of the UI in order for the user to have a simpler time figuring out how to add or modify lines or nodes in a network. The user is also able to quickly go between time-steps in the case that the time-step animation runs too fast to figure out which lines are failing on each individual time-step.

Due to the large amount of elements that pandapower provides, I had limited the SGSS to allow the user to customize the capacity of the line as well as the voltage requirement of a load or the voltage output of a generator. Although it is rather simple to add more customization options to the program, it would result is the screen being cluttered with many input boxes as well as buttons. This is something that could be changed in the future development of the SGSS by adding a pop-up window for customization options in order to

prevent clutter in the main screen of the SGSS.

Chapter 4

Technologies Used

Since the Smart Grid Security Simulator is a web application, this program consists of a front-end and back-end. The front-end of SGSS consists of the visualization of the network as well as the visualization of line failures at each time step. The front-end also contains simple data that the user can read on various nodes and lines in the network. The back-end of the SGSS consists mostly of the functions required to load networks from the pandapower package and translate them to JSON formatting in order to be sent to the front end for visualization. The back-end also holds the cascading failure calculations as well as graph calculations that are used for cascading failure.

The front-end development was built through the use of HTML, CSS, and JavaScript. These are used to describe the function and interactive elements of the program that allows the user to select networks and attack cases.

Python was used to build the back-end. Python packages pypower and pandapower were used for cascading failure simulation. pypower is useful for power related calculations while pandapower has many built-in networks that will help with the development process. The pandapower package has functions that automatically uses pypower's power flow calculations on a pandapower-formatted network. I used these functions in order to run both AC and DC power flows through networks at each time-step in the cascading failure model. After a power flow is run, the necessary calculations for line capacities are run in order to check for line failure. Through this process, the Python package pypower was not directly used in the development of the SGSS. Instead pypower was indirectly used through the use of Python package pandapower.

Chapter 5

Design Flowchart

The basic software system can be seen in figure 5.1. Once the program is launched, it initially asks the user to input a network. The user is then able to do so by selecting a network that already exists the programs database, or the user can create their own network. The network creation works by allowing the user to place their own nodes as well as edit the network capacity. After a valid network is submitted, the application loads the network to be visualized. It then prompts the user to select an attack case. Once an attack case is selected, the user is then be asked to select the location for which they want the attack to be initialized at. The program provides the user with a visualization of the cascading failure at each time step. Once the visualization is complete, the user may edit the network or choose another network to visualize. This results in the program either being rerun from the select/create step, or will result in the end of the program. In the next stages of development for the SGSS, the user will also be able to save their new or edited networks to a database which would allow users to work with networks that have been created by other people. This is included in the algorithm flowchart as well. In order to deem whether or not a network is completed, an algorithm will be developed to check for the whether parts of the network are completely disconnected from the rest.

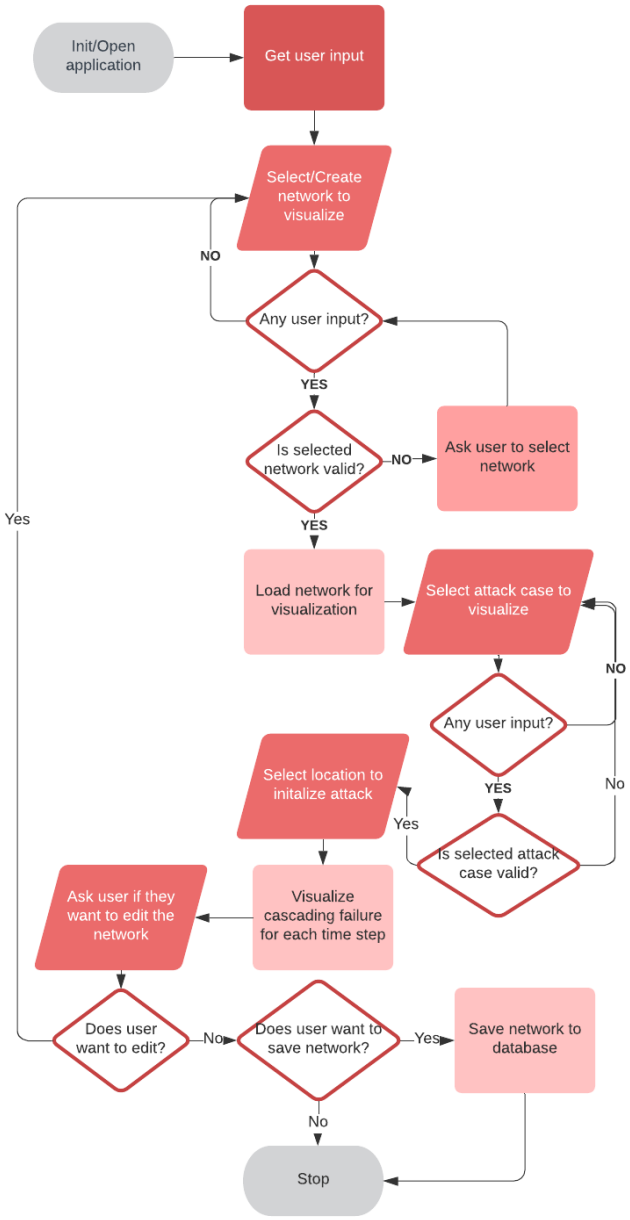

Figure 5.1: Algorithm Flowchart

Chapter 6

Design

6.1 User Interface

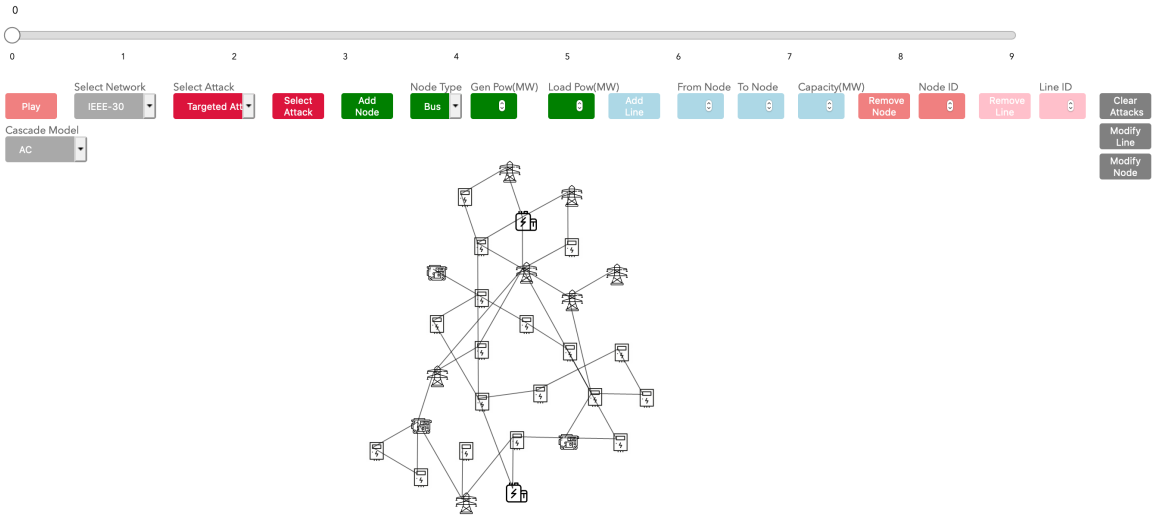

Figure 6.1: Main Screen

After launching the web application, the user will be met with the main screen of the SGSS as seen in Figure 6.1. The main screen composes of three main sections. These are the Slider, the User Input Section, and the Network Visualization.

6.1.1 Slider

Figure 6.2: Slider

The slider can be found at the top of the main screen of the Smart Grid Security Simulator as seen depicted in Figure 6.2. The functionality of the slider is very simple as there are only ten time-steps available, time-step 0 through time-step 9, in the cascading failure calculations of the SGSS. The user can move the slider between any of the ten time-steps in order to view which lines are still active in the network.

Visualization of the cascading failure effect on a given network always begins at time-step 1. At time-step 0, the user will be able to view the full network that they are testing. Time-step 1 then shows the user the network immediately after the initial attack, removing the lines from the graph. At time-steps 2 through 9, the cascading failure model is then applied and lines that fail at each time-step is removed for their corresponding time-steps.

6.1.2 User Input

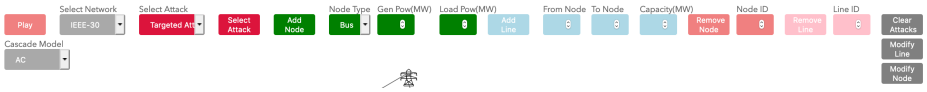

Figure 6.3: User Input

The User Input section of the Smart Grid Security Simulator contains buttons, selects, and inputs, as seen in Figure 6.3, that allows a user to customize a network. The Play button can be clicked by the user at any time in order to begin visualization of the cascading failure effect by the selected attack. This moves the slider from time-step 0 through time-step 9 in short intervals, acting as an animation so that the user can better visualize the cascading failure in the network.

The grey selects on the left-most side of the section allows a user to select which network they would like to work with as well as what power flow they would like to run in the cascading failure model. At the moment, the user can select between networks IEEE-5, IEEE-9, and IEEE-30. These networks are all pulled directly from pandapower. The user can also select between AC and DC power flows for the cascading failure model to perform during line failure calculations. The Select Attack select allows users to choose between a natural disaster attack and a targeted attack. The natural disaster attack adds all lines around the next node a user clicks into the attack case, while the targeted attack option only adds the next line that the user clicks into the attack case.

The center buttons and inputs allow a user to add and remove lines and nodes to the network that the user has selected. The green button and inputs allow a user to add a node to the network. The user can select between a bus, a generator, and a load. The generator and load options also add an initial bus connection

when added in in order to properly connect the node to the rest of the network. The user also needs to input a generator power or load power if they are adding a generator or load to the network respectively.

Similarly, the blue button and inputs allow a user to add a line to the network. The user is in charge of setting the node that the line begins at and the node that the line ends at. The capacity of a line is also inputted directly by the user.

The pink buttons and selects allow a user to remove nodes or lines according to the line or node id that the user inputs. This allows a user to more easily remove the line or node that they intend to remove rather than having the potential of accidentally clicking an alternative line or node.

Finally, the user can clear the attack case, or modify a specific line or node. At the current state, the modification process uses the existing inputs for line and node ids, as well as the power inputs and line capacity inputs.

This section is planned to be updated into a pop-up window in order to reduce the clutter when more modification options are added, as well as add clear instructions for what each part of the section does.

6.1.3 Network Visualization

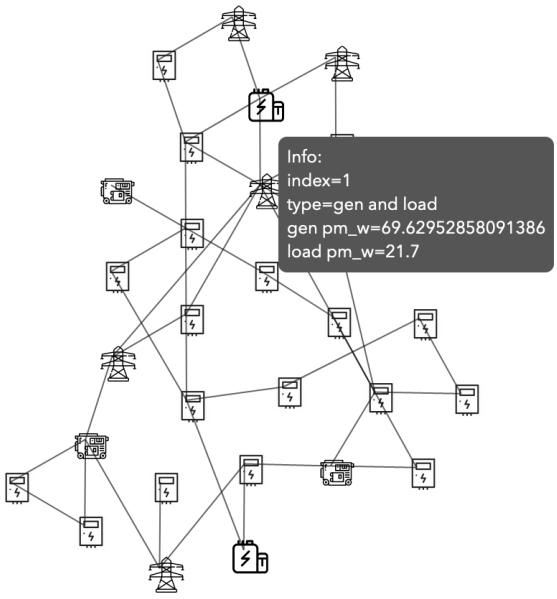

Figure 6.4: Network Visualization. Source: Icons made by Freepik, smalllikeart from www.flaticon.com.

Figure 6.4 shows the IEEE-30 network provided by pandapower. The network consists of all four types of nodes that are allowed in the SGSS. These are the bus, the generator, the load, and the generator and

load. As seen in the figure, hovering over a node or line presents the user with a small pop-up showing basic information on the element that the user is hovering over.

In the case of the figure, the user is hovering over a generator and load. The node id is 1 and the node's corresponding generator power and load power is given in megawatts.

To provide more visual simplicity, the visualization of a power grid is shown like that of graph theory. Rather than showing every bus and generator/load connected to it, SGSS only shows the combination of a bus and generator as a generator in the front-end. This goes for loads connected to buses as well. However, calculations for the power grid still consider all buses involved. Nodes in the visualization section can be dragged around by the user in order to rotate the graph or change the positions of specific sections of the graph. This allows a user to work around nodes that are too close together as well as better visualize cascading failures when lines are initially overlapping.

It is important to note that the length of a line is not shown in the simple network depiction given by SGSS's visualization of a network. Since nodes can be dragged around by the user, it is difficult to truly implement the lengths or distances of lines and nodes in the visualization process.

6.2 Cascading Failure Model

To calculate line capacities I used the formula given in equation 6.1 below, as seen in [5]

$$c = (1 + \alpha) \times max(|f_l|, \overline{f}) \tag{6.1}$$

To begin, the line capacities are calculated with the line tolerance (set to 1) plus 1 times the max between the average initial magnitude of all line flows and the magnitude of active power at the receiving side of the line. Essentially, the line capacities are calculated to 2 times the max of those values.

Next, the network is temporarily turned into a simple graph of nodes and lines to check if all the nodes are connected. If not all nodes are connected to one another, then the network is separated into sub networks in order to perform further calculations. Once this is complete, the power generated by generators in the grid are then re-scaled based on the total load divided by the total generated power. This is known as Generation Ramping which is described in [4]. And finally, a DC or AC power flow is run through each network. After the power flow, lines are checked to see if they exceed their calculated capacities, and if they do, the line is failed. The data is saved for each time-step and sent back to the front-end for visualization.

In [5] the authors used a different supply and demand balancing rule. This rule being, shedding and curtailing. They did this by checking if the total power supply was greater than the total power demand. If it

was, then the active power outputted by generators were curtailed. If not, then load shedding was performed to match the total power supply. This differs from the Smart Grid Security Simulator's implementation of re-scaling power generated in order to match the power demand. The authors' calculation for line failure was implemented using the deterministic rule of power flow in lines exceeding the line capacity similarly to the deterministic line failure used in the cascading failure model of the SGSS.

The alternative rule mentioned in [5] is separating and adjusting. This rule begins by separating the generators from the grid from smallest to largest until a removal results in a shortage of supply. After separation, the output of the largest supply node is reduced to meet the demand. This rule for supply and demand balancing is planned to be added in the next stages of development for the SGSS.

Testing for the cascading failure model in SGSS was compared to the test results for the IEEE-30 network as given in [5].

Chapter 7

Testing

7.1 Testing Procedures

In order to test the cascading failure model that I had implemented for the Smart Grid Security Simulator, I decided to work with the IEEE-30 network case file provided by pandapower. This decision was due to the clear test results for the same network provided in [5] that allowed for an easy comparison in order to check the accuracy of SGSS's cascading failure model.

In order to test the network, I decided that single-line failure would be the most practical approach. Although the journal [5] provides both single-line and double-line failures, the time it would take to run all the tests for both would take far too long on the current implementation of the SGSS.

7.2 Test Results

Through testing the failure of a single line of all lines in IEEE-30, I found the following information.

The failure of line 4 resulted in line 8 failing in the following time-step, and lines 0 and 7 failing in the 4th time-step in both AC and DC power flows. A total of 3 lines failed after the initial attack.

The failure of line 7 resulted in line 8 failing in the following time-step, and line 0 failing in the 4th time-step in both AC and DC power flows. A total of 2 lines failed after the initial attack.

The failure of line 9 resulted in lines 39 and 40 failing in the following time-step. In an AC power flow, only line 0 failed in the 4th time-step, while in a DC power flow, both lines 0 and 34 failed on the 4th time-step. A total of 3 to 4 lines failed after the initial attack.

The failure of line 28 resulted in lines 26 and 27 failing in the following time-step. In a AC power flow, lines 0, 29, 30, and 35 failed in the 4th time-step and lines 1, 3, and 34 failed in the 5th time-step. In a DC power flow, lines 29, 30, and 35 failed in the 4th time-step and lines 0, 3, and 34 failed in the 5th time-step.

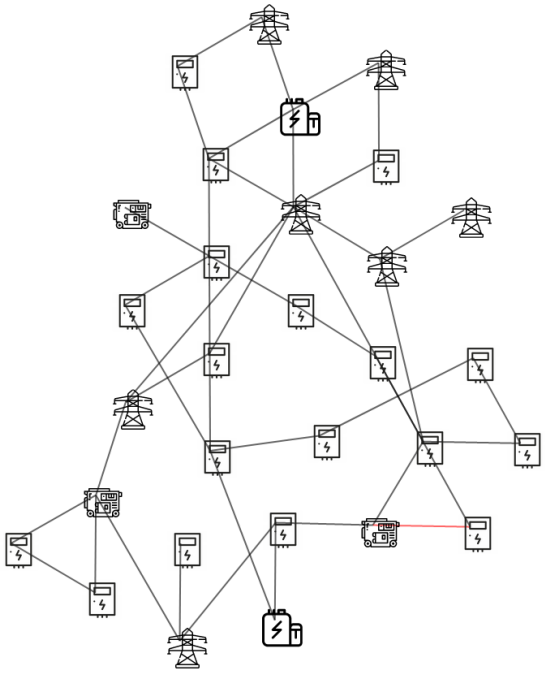

Figure 7.1: IEEE-30 AC Time-step 0

Finally, line 0 failed in the 6th time-step. A total of 9 lines failed after the initial attack.

Figure 7.1 shows time-step 0 for an initial attack case of line 28. As shown in the figure, the line that is selected for the attack case is highlighted red so that the user is able to clearly view which line they have selected. Figure 7.2 shows time-step 5 for the initial attack case of line 28. In this figure, lines 0, 1, 3, 29, 30, 34, and 35 are missing to show that these lines have failed in the time-steps between time-step 0 and time-step 5.

The lines with the largest vulnerabilities should be lines 1, 9, then 28, 30, 31, 36, 40, and 41 according to [5].

However, in the testing done using the cascading model I had implemented, lines with the largest vulnerabilities came out to be lines 28, 9, then 4, 7, 29, 31, 34, and 35. There are some similarities between the SGSS tests and the paper's results. Even then, the order of the vulnerabilities are still off. This difference is likely due to the difference in implementation of power generation scaling compared to the shedding and curtailing used by the paper's authors. Given the remaining time to develop, I will revisit this issue to continue testing and improve the accuracy of the SGSS. However, many conclusions can still be made with the current cascading failure model implementation.

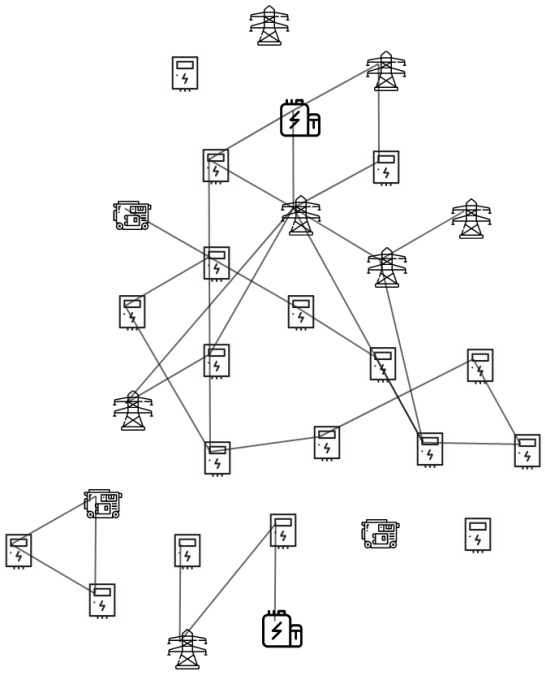

Figure 7.2: IEEE-30 AC Time-step 5

Chapter 8

Societal Issues

8.1 Ethical

As a teaching tool, the Smart Grid Security Simulator is designed to educate users on the functionalities of a smart grid and how they respond to various attacks on a network. With this knowledge, users can help improve on existing power grids across the world in order to help prevent events such as the Northeast Blackout from being as severe. This could result in many lives being saved as well as prevent many families and businesses from losing essential power that is needed in their day-to-day lives.

8.2 Technological

The Smart Grid Security Simulator helps advance the current technology in cascading failure prevention. Due to the lack of existing applications that allow for simple modifications of a network, the development of the SGSS helps those learning about smart grids and cascading failures to better understand how changes to a network can improve or reduce the security of the network.

8.3 Social

The Northeast Blackout of 2003 resulted in approximately 50 million being affected and many of those losing power for up to 2 days. This sort of power less resulting from a cascading failure due to a lack of security in power grids leads to many social impacts. Without being able to contact families or individuals who suffer from the lack of power, friends or families of the people impacted are unable to assure that the people they care about are safe from the disaster. Additionally, those affected are unable to contact help when needed. The improvements in security of power grids across America, and eventually, across the world can lead to the prevention of a massive blackout. This can result in the power-security for everyone who is connected to

a power grid across the world. Although smaller communities may still experience blackouts due to targeted attacks and natural disasters, the scale of the blackout would be nowhere near as large as the Northeast Blackout.

8.4 Economic

Large-scale blackouts across the world results in massive strains on the economy of the area that was impacted. The Northeast Blackout alone costed around 6 billion dollars for repairs, with an unknown cost as to how much it had costed the families and businesses in the area. The prevention of larger blackouts could save countries a large sum in repair costs, which could instead be put towards the strengthening of security for their power grids. Although the application itself does not cost much to upkeep as well as having no existing plan to charge for usage of the Smart Grid Security Simulator, the end-goal is to help provide more teaching tools to those learning about smart grids and cascading failure effects.

8.5 Health and Safety

Although there are no direct health and safety concerns that are brought up by the SGSS. The accuracy of the final product of the SGSS is important for the safety of power grids. Since it is built as a teaching tool, the knowledge gained from using the SGSS is meant to be used to help improve on existing and future smart grids. If the accuracy of the application is far from accurate, it may lead to failures in power grids in the future. This would have a negative impact on the health and safety of people who rely on these power grids.

8.6 Manufacturability

Since this project is developed as a web application, there is no need to continue the reproduction of the Smart Grid Security Simulator. Additionally, there is little to no upkeep cost for it.

8.7 Usability

The SGSS is designed to have a minimal and simplistic view of how cascading failure in power grids work. In reality, a bus in the 30-bus network can be connected to a generator, a load, or both a generator and a load. However, in the SGSS visualization of a network, the bus is combined with the generator and/or load in order to be visualized as a single component. This is to optimize the user's experience with the application in order for networks at a larger scale to be much more manageable for users learning about cascading failures.

The UI is designed using color-coding for buttons and text input so that a user can more easily find which inputs to use when adding or modifying nodes and lines in a network. Additionally, when hovering over a specific node or line, information about the element is displayed. This allows users to distinguish which node or line they would like to edit or remove.

In its current stage, the Smart Grid Security Simulator web application does not take into account for accessibility by everyone. There are no text-to-speech options for people with visual impairments available on the application and the current text size used may be difficult to view as well. The accessibility of the application is extremely important and I plan to make many improvements on the accessibility-aspect in the future development of the project.

8.8 Sustainability

The Smart Grid Security Simulator is a web application that will be viable and useful until large changes to the implementations of smart grids are made. Additionally, since there is a negligible upkeep cost for the SGSS, it has little to no negative impacts on the environment and economy. The knowledge that can be passed along to users of the SGSS may help benefit power grids and the development of smart grids in the future.

8.9 Environmental Impact

Since the SGSS is completely developed as a web application, the development process of the Smart Grid Security Simulator has had no impact on the environment. The knowledge gained from the use of the SGSS may lead to more efficient power grids which may lead to small improvements on the environmental impact of power grids.

Chapter 9

Conclusion

9.1 Obstacles Faced

Although I did not end up facing many obstacles that I hadn't anticipated facing, there were a few issues that resulted in a delay in the development process of the SGSS. The main obstacle that I faced was my initial lack of understanding of power grids and cascading failure. Since I had not taken many courses about the topic prior to starting the project, I had to set aside a lot of time in order to just do research. I spent the majority of Fall quarter researching on how smart grids function as well as how a targeted attack or natural disaster could effect a power grid. After solidifying some basic understanding in smart grid functionality as well as attacks on power grids, I had decided to dive right into the development process of the SGSS.

This led to the first major setback that I had encountered. I had initially began writing the cascading failure model to do checks on nodes that were directly connected to the node of the initial attack. This model would spread out as it continued to check more nodes as the failures spread out. However, after checking my results against the results of existing studies, I found that my cascading failure model was completely off. In order to proceed, I had to do much more in-depth research on how cascading failure worked. This eventually led to the current model of checking the entire grid after each grid modification, as nodes on the opposite side of the power grid from the initial attack may also fail in direct correlation to the initial line failure.

Another obstacle that I faced throughout the research and development of the SGSS was working on the project individually. It was much easier to manage how different components worked with each other for the SGSS since I developed all of the individual components. However, I was unable to refer to other members of a group when I ran into a roadblock in development or when I didn't fully grasp a concept during the research phase. This made it difficult, as a lot of the time spent developing the project was spent debugging issues that could have easily been solved. Additionally, pandapower is not a very popular Python package, which made

debugging pandapower related problems much more time-consuming.

9.2 Lessons Learned

Going off of the previous section, the most important lesson that I had learned was the importance of research before development. After getting somewhat impatient with the research process and diving directly into the development process, I found myself wasting many hours of development and debugging on issues that I simply had wrong from the get-go. If I had stuck with the initial plan that I had setup with my advisor, Dr. Xiang Li, I would have been able to fast-track the beginning stages of the development process simply by fully completing the research part before starting. On top of this, I should have planned for more research to begin with, as I did not consider the amount of time that it would take to adequately learn how to work with pandapower functions as well as manipulate pandapower networks.

During the development stage of the SGSS, I also learned of the importance of documentation as well as the readability of code. At the beginning stages of development, I failed to properly document functions that I was adding since I initially believed it was not necessary if I was the only one working on the project. However, this caused problems later down the line when I had issues figuring out what each of the functions I had written were doing as well as the variable for the input and output of the functions. As I had begun documenting my work, I found that is streamlined the development process as I added more and more functions but was still able to keep track of what each one of them did.

Finally, in the process of working on the project as a one-person group, I learned many of the benefits and drawbacks of working alone. I was able to fully manage when each of the individual components of the SGSS were completed as well as being able to work on my own time schedule. However, I found it difficult to go back and debug the code without the help of fellow group members. This was especially difficult, as I found it harder to keep track of every element that was being changed at each time-step as well as checking whether or not that element should have been changing or not. Additionally, the testing phase of the cascading failure model of the SGSS was extremely taxing as it required me to individually test each single-line failure in the 30-bus network as well as keeping track of each of the line failures.

Although I was able to get feedback from a few testers for the UI and the functionalities of the SGSS, I do not believe that I had gotten enough people to test the web application out. Since the SGSS is meant to be a learning tool in the end, I should have spent more time getting others to test out the SGSS to see if they were able to understand how to use it without being given verbal instructions.

9.3 Future of Project

When it comes to the continued development of SGSS, I plan on improving the accuracy of the cascading failure calculations. This can be done by adding more models that the user can select from such as shedding and curtailing. Additionally, many different types of nodes can be added to SGSS to cover all of the data structures and elements that are given in pandapower. These includes motors, shunts, transformers, switches, and many more. This would allow users to fully customize a network.

Building upon the requirements that have not been met, the SGSS can also be improved upon to support the creation of a new network as well as the saving/accessing of networks that have been implemented or saved by the user.

Finally, strategies can be implemented into the SGSS to allow the application to provide users with the best-calculated improvements that can be made on a given network. These improvements would consider which lines are most important to be improved upon as well as suggestions on the parameters of newly added nodes.

The addition of these would also require an improvement of processing time in order to handle much larger networks. As the current implementation of the SGSS already struggles with handling larger networks.

When all of the improvements suggested in the previous slide are implemented, the SGSS will be able to provide power companies with accurate methods on how to best improve on a given power grid based on the budget they are allowed. With the improvements of power grids across America, blackouts can be largely avoided when it comes to targeted attacks as well as natural disasters. With the avoidance of these blackouts, there will be less interruption in everyone's daily lives as well as the avoidance of many deaths such as the lives lost due to the Northeast blackout. Power companies will also be able to test networks that have yet to be developed under various attacks in order to determine what improvements can be made before installing new networks across the globe.

Chapter 10

References

[1] H. Haes Alhelou, M. Hamedani-Golshan, T. Njenda, and P. Siano, “A Survey on Power System Blackout and Cascading Events: Research Motivations and Challenges,” Energies, vol. 12, no. 4, p. 682, Feb. 2019.

[2] J. R. Minkel, “The 2003 Northeast Blackout–Five Years Later,” Scientific American, 13-Aug-2008. [Online]. Available: https://www.scientificamerican.com/article/2003-blackout-five-years-later/. [Accessed: 31-May-2021].

[3] A. Muir, and J. Lopatto, “Final report on the August 14, 2003 blackout in the United States and Canada : causes and recommendations,” Canada: N. p., Apr. 2004. Web.

[4] J. Yan, Y. Tang, H. He, and Y. Sun, “Cascading Failure Analysis With DC Power Flow Model and Transient Stability Analysis,” IEEE Transactions on Power Systems, vol. 30, no. 1, pp. 285–297, 2015.

[5] H. Cetinay, S. Soltan, F. A. Kuipers, G. Zussman, and P. Van Mieghem, “Comparing the Effects of Failures in Power Grids Under the AC and DC Power Flow Models,” IEEE Transactions on Network Science and Engineering, vol. 5, no. 4, pp. 301–312, 2018.