Detecting Network Vulnerabilities Through Graph Theoretical Methods

Patrick Cesarz
Villanova University

Gina-Maria Pomann
The College of New Jersey

Luis de la Torre
University of California, Davis

Greta Villarosa
The College of William and Mary

Tamara Flournoy
University of Michigan

Ali Pınar
Lawrence Berkeley National Laboratory

Juan Meza
Lawrence Berkeley National Laboratory

October 18, 2007

Abstract

Identifying vulnerabilities in power networks is an important problem, as even a small number of vulnerable connections can cause billions of dollars in damage to a network. In this paper, we investigate a graph theoretical formulation for identifying vulnerabilities of a network. We first try to find the most critical components in a network by finding an optimal solution for each possible cutsize constraint for the relaxed version of the inhibiting bisection problem, which aims to find loosely coupled subgraphs with significant demand/supply mismatch. Then we investigate finding critical components by finding a flow assignment that minimizes the maximum among flow assignments on all edges. We also report experiments on IEEE 30, IEEE 118, and WSCC 179 benchmark power networks.

1 Introduction

The electric power grid network is susceptible to power outages that may potentially cause severe blackouts. A striking example is the August 14, 2003 blackout in US northeast and Canada, which affected an estimated 50 million people, causing over $6 billion in damage, and leaving many areas without power for two days [3]. Because of the size and complexity of the power network, it is extremely difficult to quickly identify the power lines

that cause the most damage when cut from the network. Thus, it is imperative to be able to pinpoint the most critical power lines in a network so that vulnerable components may be fortified in order to preempt disasters. As such, the purpose of this research is to formulate and analyze efficient methods for detecting critical lines in a power network. While the focus of our work is power networks, our techniques are applicable to other systems such as the transportation and communication systems. For example, Auro et. al. investigate network inhibition in the context of computer networks, primarily for finding vulnerabilities of public computer networks that may have low bandwidth connections and low connectivity areas [1]. In these networks, finding vulnerable, loosely connected components can prevent significant revenue loss to internet service providers and commercial websites from major bandwidth loss.

The principal problem that we investigate is the problem of network inhibition. This problem seeks to maximally inhibit the functioning of a network, by finding a subset of edges of a network of some (usually small) fixed size of connections that when removed from a power network together maximally inhibit the total flow through the network. The network inhibition problem is NP-complete [5], thus we need to resort to heuristics for a practical solution, since time for an optimal solution can scale exponentially with the size of the network.

An approximation of the network inhibition problem is the inhibiting bisection problem [6]. The inhibiting bisection problem looks for a bisection of a network that maximally inhibits its function. In contrast, the solution to the network inhibition problem does not necessarily define a cut in the network. The problem seeks to minimize the number of lines cut in a network while maximizing the supply/demand equilibrium between the two sections. Inhibiting bisection finds a bisection that places generators and consumers on different bisections as much as possible. However, it is constrained by the number of edges that are between these components. The inhibiting bisection problem is also NP-complete [6].

We investigate a polynomial-time relaxation of inhibiting bisection in order to estimate solutions to the problem and find loosely connected components of a network. In the following report, we investigate efficient ways of approximating inhibiting bisection utilizing the relaxed version. In addition, we attempt to use inhibiting bisection as a way to find the most critical connections in a network and consider an alternative constrained linear optimization problem to rank edges in order of criticality.

2 The Inhibiting Bisection Problem

We model electric power grid networks as undirected graphs. A graph $G = (V, E)$ is defined by a set of vertices V and edges E connecting vertices. Vertices model consumers and producers of power, and edges model

the connections between consumers and producers. Both vertices and edges have weights assigned to them. Vertex weights correspond to supply and demand values, with positive values indicating supply and negative values indicating demand. Edge weights denote edge capacities, i.e. the maximum amount of power that can flow through the edge. Our units of power are normalized values. In Figure 1, we see a small network with edge capacities and supply and demand values. In this figure, generator vertices are represented as diamonds, and consumer vertices are represented as circles. Note that supply and demand values must add up to zero, as total supply equals total demand.

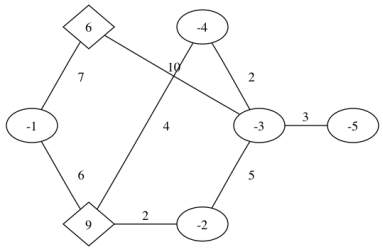

Figure 1: A small network represented as a graph

Inhibiting bisection problem has been introduced by Pinar et. al. [6], to identify loosely-connected components of a graph with significant generation/load mismatch. That is, with inhibiting bisection we can find a subset of the network that can be disconnected from the rest with only few edge removals. To define the inhibiting bisection problem, we define a few terms. Given a graph $G = (V, E)$, a bisection is a partition of V into two disjoint sets, V_1 and V_2. The cut set $C(V_1, V_2)$ is the set of edges in E that pass between the bisection, i.e. $C(V_1, V_2) = \{(i, j) : i \in V_1, j \in V_2\}$. Note that V_1 and V_2 need not be connected subgraphs. Define w_i as the weight of vertex v_i. The weight of a set of vertices is the sum of the weights of all the vertices in the set, i.e. $W(V_1) = \sum_{v_i \in V_1} w_i$). Given a bisection of V into V_1 and V_2, the imbalance of the bisection is the difference in the weights of V_1 and V_2, or $|W(V_1) - W(V_2)|$. Finally, the cutsize of a bisection is defined to be the number of edges in the cutset, or $|C(V_1, V_2)|$. The inhibiting bisection problem is defined as follows:

Given a graph $G = (V, E)$ and a cutsize constraint of B, find a partition of V into V_1 and V_2 that maximizes the imbalance $|W(V_1) - W(V_2)|$ subject to the cutsize $|C(V_1, V_2)| \leq B$.

Alternatively, the problem may be defined as finding a minimum cut size subject to a minimum constraint on the imbalance value.

2.1 Relaxed Version of Inhibiting Bisection

Because the inhibition bisection problem is NP-complete, we look for an approximation to the problem. Hence, we investigate of the relaxed version of

inhibiting bisection. The relaxed version of the problem uses a dual objective function, and takes an input parameter, ϵ. Given a value of $\epsilon \in [0, 1]$, the relaxed inhibiting bisection is the following minimization problem:

$$\min \epsilon|C(V_1, V_2)| - (1 - \epsilon)|W(V_1) - W(V_2)|. \quad (1)$$

In this formulation, both the cutsize $|C(V_1, V_2)|$ and the generation-load mismatch $|W(V_1) - W(V_2)|$ are in the objective. The choice of ϵ weights the importance of the objectives. A value of ϵ closer to 1 gives more importance to minimizing cutsize than maximizing imbalance, and a value of ϵ closer to 0 indicates that maximizing imbalance bears more importance than minimizing cutsize. As ϵ increases from 0 to 1, both cutsize and imbalance decrease monotonically.

3 Inhibiting Bisection Approximation

In this section, we will show how the relaxed version of the inhibiting bisection problem can be solved using a maximum flow solver, as originally described by Pinar et al. [6]. Given a graph $G = (V, E)$ with edge capacities and two specified vertices s (source) and t (terminal), the problem of maximum flow is to find the maximum amount of flow that can be carried from the source to the terminal constrained by the capacities of the edges. A minimum cut of a graph is a bisection of the graph such that s and t are on different partitions and the sum of the capacity on the edges in the cut set is minimized. It is a well-known duality result that the minimum cut value of a graph is equivalent to the maximum flow value.

To formulate relaxed inhibiting bisection as a maximum-flow minimum-cut problem, we use the following transformation.

- Assign a capacity of ϵ to each edge in the given network.
- Add vertices s and t, and connect each supply vertex v_i to s by an edge with a capacity of $2|(1 - \epsilon)w_i|$, and connect each demand vertex v_i to t, by an edge with a capacity of $-2|(1 - \epsilon)w_i|$.

Pinar et al. [6] proved that the minimum cut solution is the bisection corresponding to the optimal solution of the given relaxed inhibiting bisection problem. In Figure 2, we see the graph of Figure 1 transformed to a max-flow min-cut problem for solving the inhibiting bisection problem with $\epsilon = 0.5$.

We used the implementation of Boykov and Kolmogorov [2] as the max-flow solver. Recall that both the cutsize and the imbalance decrease monotonically with increasing ϵ. We can use this observation to approximate the original version of the inhibiting bisection problem. We optimize either imbalance or cutsize with a constraint on the other by iteratively solving the relaxed version of the inhibiting bisection problem with different values of ϵ. The iterative process helps us determine the maximum imbalance for a

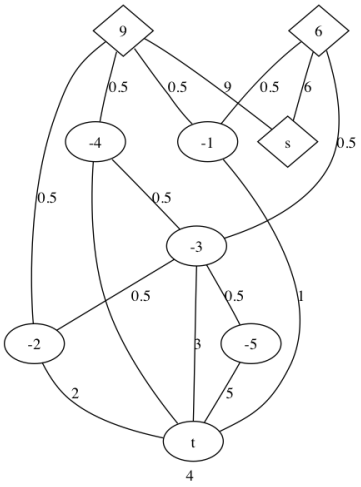

Figure 2: Figure 1 reformulated as max-flow min-cut with $\epsilon = 0.5$

desired cutsize, or the minimum cutsize for a desired imbalance. We explored two methods that would perform the iterative search for ϵ values: a binary search method and a secant method.

The binary search method and secant method are both iterative methods used to search for where a given function attains a certain value (usually zero). In this case, we are interested in finding a value of ϵ for which the objective function returns a given cutsize bound. In the binary search method, the search interval is halved after every iteration. This halving is performed based solely on the cutsize computed at the current value of ϵ. The secant method, as described in [4], takes into account how far off the cutsizes are from the cutsize constraint to compute the next ϵ value by drawing a secant line from the points at the boundary of the current search interval. Since the secant method uses more information to update ϵ, we expected this method to converge faster than the binary search method. This, however, was not the case in our experiments.

We tested both search methods to solve the inhibiting bisection problem for all feasible cutsize constraints of a given graph. We used the IEEE 30 bus graph, the IEEE 118 bus graph, the WSCC 179 graph, and a smaller example graph (with 7 vertices and 8 edges). We first considered the number of iterations each algorithm took for each cut size constraint. In the small example, both algorithms worked equally well. In the 30 bus graph, the program solves the inhibiting bisection problem for 18 cutsize constraints. The binary search terminated in fewer iterations than the secant method in 5 of these cutsizes, and the secant method performed better in 8 cutsize constraints. For the 118 bus graphs, the binary search was better in 31 cutsize constraints, and the secant method was better in 33 test cases. In the WSCC 179 graph, the binary search method performed better in 34, and the secant method performed better in 54 cases. Based on this analysis, it is

hard to declare a clear winner.

We then considered the total number of iterations each method uses to find the cutsizes for all feasible cutsize constraints. These data are summarized in Table 3.

	Bisection	Secant
IEEE 30	138	146
IEEE 118	501	574
WSCC 179	801	0

Table 1: Number of iterations for bisection and secant methods

It seems that as the size of the graph increases, the difference in the two method's performances also increases. If this pattern continues, we expect the secant method to perform worse than the binary search method for larger graphs.

Each algorithm was also timed for the 118 graph and the 179 graph. Table 3 displays the time it takes the program to run in seconds.

	Bisection	Secant
IEEE 118	.06	.08
WSCC 179	.09	.10

Table 2: Elapsed time in seconds for secant and bisection method for finding all cutsizes

Both algorithms can use the exact same stopping criteria. An easy stopping criteria would be the number of iterations (e.g. we used 20 as an upper bound on the number of iterations). However, this condition can be improved. Given a bound B, if we find a bisection with cutsize equal B for the relaxed inhibiting bisection problem, then we can conclude that the optimal answer is found, and the algorithm can terminate, since any other solution will either violate the constraint on the cutsize, or will provide a solution at most as good as the present one. The following theorem formalizes this claim.

Theorem 1 *In any graph, two optimal solutions to the relaxed inhibiting bisection problem have the same cutsize if and only if they have the same imbalance.*

Proof: We prove that equal imbalances imply equal cutsizes, and the converse is proved similarly. Let us call the objective function of the relaxed inhibiting bisection problem $f_\varepsilon(C, i)$ where ε is the parameter, C is the size of

the set of the edges cut (cutsize) and i is the imbalance. For a given $\varepsilon \in [0,1]$ we are interested in finding the cutsize and imbalance that minimizes

$$f_\varepsilon(C,i) = \varepsilon(C) - (1-\varepsilon)i.$$

Let $\varepsilon_1, \varepsilon_2 \in [0,1], \varepsilon_1 \neq \varepsilon_2$, (C_1, i) be the minimizer of f_{ε_1}, and $(C_2$,i) be the minimizer of f_{ε_2}. That is two different values of ϵ yield the same imbalance. We need to show $C_1 = C_2$ in this case. Assume the contrary, and without loss of generality, let $C_1 > C_2$. Since (C_1, i) is a minimizer of f_{ε_1}, we have

$$\begin{aligned} f_{\varepsilon_1}(C_1,i) &\leq f_{\varepsilon_1}(C_2,i) \\ \varepsilon_1(C_1) - (1-\varepsilon_1)I &\leq \varepsilon_1(C_2) - (1-\varepsilon_1)I \\ \varepsilon_1(C_1) &\leq \varepsilon_1(C_2) \\ C_1 &\leq C_2, \end{aligned}$$

which contradicts the assumption that $C_1 > C_2$. □

Observe that the converse is also true: bisections with the same imbalance have equal cutsizes. This fact allows the algorithm to terminate with an optimal solution, once it finds a bisection with cutsize equal to the cutsize constraint.

If the search interval is sufficiently small, the algorithm stops. Our implementation terminates if the interval is less than 10^{-6}. Note that this gap is the reason why we have an approximation algorithm, not an exact solution, and the gapsize is a bound on the accuracy of our approximation. All these stopping conditions are conventional for any binary search or secant method problem, but due to the nature of cutsize as a function of ϵ, we can add another stopping condition. From Figure 3 it is clear that this function is piecewise constant (it is a step function) and monotonically decreasing. As ϵ increases, minimizing cutsize is more important than maximizing imbalance. This explains why the function is monotonically decreasing. It is piecewise constant because in any graph there are only a finite number of minimum cuts. In the 30 bus graph, for example, minimum cuts have cutsizes of 1, 5, 9, 11, and 14 edges.

Due to the nature of the graph, the secant line grows steeper after each iteration. If the secant method (or bisection method) are close to converging, then the secant line will be between two steps. At this point in the algorithm, the difference in cutsizes between the two points is constant, but the search interval for ϵ is decreasing. Thus, the secant line grows steeper after each iteration. If this line is sufficiently steep, then there is less of a chance that another smaller step will exist in the interval containing the secant line. In our experiments, the algorithm terminated if the absolute value of the slope is greater than 1000. For the WSCC graph, however, this slope had to be increased to 10,000 in order to find all the cutsizes.

In theory, the secant method should converge faster than the binary search method. The secant method converges slower, however, because of the nature

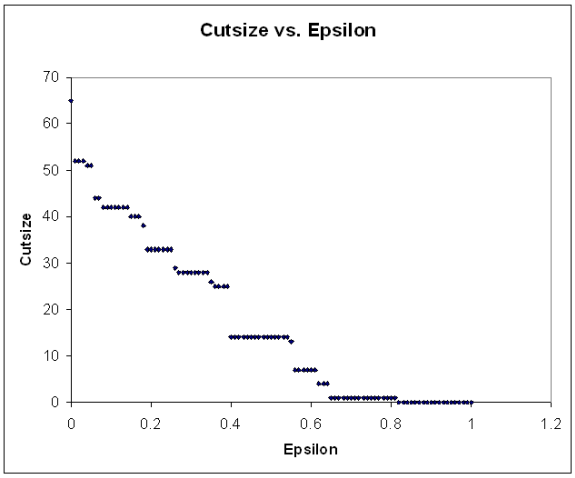

Figure 3: Cutsize vs. Epsilon For IEEE 118 Bus

of the cutsize vs. ϵ function. The secant method performs better for continuous functions, and this function is not continuous. In the binary search method, the search interval is always halved, regardless of the nature of the function, as long as the function is monotonic.

Both functions still have limitations. Both algorithms can only estimate values of ϵ to a certain precision. If one of the steps in the cutsize function is sufficiently small, then neither algorithm will detect it, and one of the feasible counts will be missed. This problem can be minimized if the algorithms were allowed to run for more iterations. But no matter how many iterations the program runs, it is always possible to construct a graph where the length of the step is sufficiently small to go undetected.

4 Frequency Analysis

We used our techniques to find a bisection for each possible cutsize and list edges in these bisections. We define the frequency of an edge as the number of times it occurs throughout all bisections. The hypothesis regarding frequencies is that the higher the frequency of an edge the higher the criticality of that edge to network inhibition. The criticality of an edge is the extent to which it inhibits the total flow through a network. We tested our hypothesis on three benchmark networks: The IEEE 118 bus network which has 118 nodes and 179 edges, the IEEE 30 bus network which has 30 vertices and 41 edges, as well on the WSCC 179 network which has 179 nodes and 222 edges.

4.1 IEEE 118 Network Analysis

In the case of the IEEE 118 network, the edge (8,9) is in 100% of the bisections and causes the most damage when removed alone. The flow loss due to

removal of this edge was 4.500014, which encompasses 14.02% of the flow available to the network. Figure 4 shows the flow vs. frequency of each edge when removed individually. Edge (8,9) is the obvious outlier with a frequency of 17.

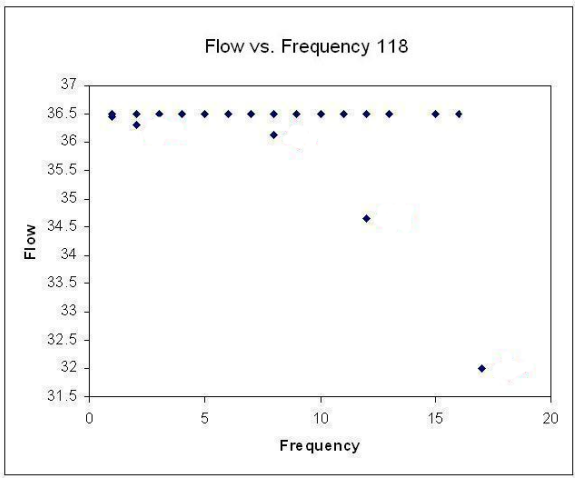

Figure 4: Flow vs. Frequency of the IEEE 118 Bus System

When any edge in the network is cut in conjunction with edge (8,9) the combination inhibits the flow significantly. However, the variation of flow in those inhibitions are within the interval of [4.50, 4.86]. There exist only two cases in which the variation of flow is higher than 4.500019. This occurs when edge (8,9) is cut with edge (110,111) or edge (86,87). We observe that edge (110,111) is the sole connection to a generator of size .36 units. Also, edge (86,87) is the sole connection to a generator of size .04 units. The variation in flow when edge (8,9) is coupled with any edge other than edge (110,111) or edge (86,87) is 0.00009. It can then be concluded that for this network unless edge (8,9) is cut in conjunction with edge (110,111) or edge (86,87) the flow will not be inhibited significantly in comparison to the cutting of edge (8,9) alone.

We have not detected a bisection of size two, however, both edge (110,111) and edge (86,87) have high frequencies and are within 50% of the bisections. Despite the fact that the pairs {(8,9), (86, 87)} and {(8,9), (110, 111)} do not appear in the cut set for a cutsize of two, the program uses the frequencies to identify them as critical to inhibition of flow. Using this measure of criticality we can decrease the number of edges considered when looking for the most critical edges. There are 167 combinations of edges with (8,9). Restricting the search to the edges that appear in the frequency list reduces the number of edges to 76. Within this set we find that the only two edges that cause significant loss of flow when coupled with edge (8,9) are found within the upper half of the frequencies. Limiting the search to those edges that are found within 50% of the frequencies reduces the number of combinations to 28. Therefore, the utilization of this deduction could possibly decrease

the search time significantly and produce a solution in a reasonable time period. This method of search for the combination of two most damaging edges reduces the number of combinations to be considered by 83.2%.

4.2 IEEE 30 Network Analysis

The IEEE 30 Network follows a similar pattern. The edge (12,13) is found within 100% of the bisections, connects the second largest generator (1.85 units) and its removal inhibits the flow by 22.52%. Figure 5 exhibits this. The outlier on Figure 5 is the edge (12,13) with a frequency of 6.

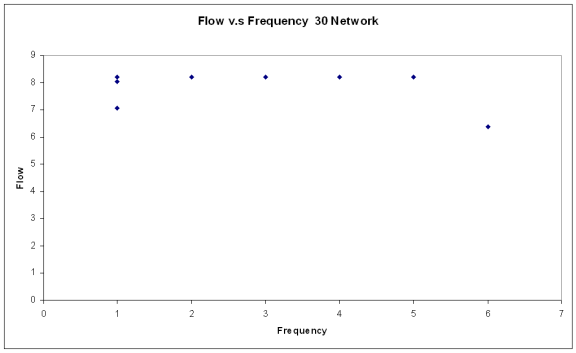

Figure 5: Flow vs. Frequency of the IEEE 30 Bus System

The cutsize of one separates the second largest generator from the graph. The largest generator in this network is of size 1.964 and accounts for 23.9% of the generation available to the network. Another bisection isolates two generators, including the largest generator, with a combined generation of 3.14. These generators can be identified as 1 and 2 on the figure in Appendix 4. This network differs form other networks we have studied, in that edge (12,13) is the only edge that serves as the sole connector for a generator. Therefore, the bisections clearly detect the edges that isolate the larger generators. A bisection of size 5 causes a total loss of 4.99 units or 60.75%. All of the edges cut within the bisection of cutsize five have over 60% frequency. When evaluating all edges connected to generators, we can only state that those edges found within more than 16% of the bisections are those connected to generators. However, all edges connecting generators to the network are within the frequency list. By concentrating only on the frequency list of edges one can reduce the search for critical edges by 46.34%.

4.3 WSCC Network Analysis

It is important to note that in the WSCC Network all of the generators are only connected by a single edge as can be viewed in Appendix 4. In the WSCC network the edge (75,76) connects the largest generator of weight 93.965, and when removed, it inhibits 25.88% of the flow. The graph of flow

vs. frequency can be viewed in Figure 6, which displays one extreme outlier which is edge (75,76).

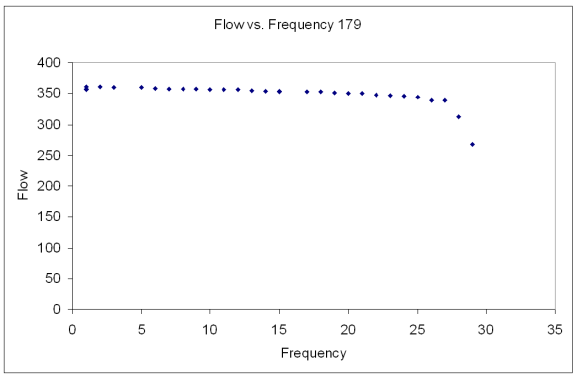

Figure 6: Flow vs. Frequency of the WSCC 179 Bus System

A valuable observation is that for any network removing a large generator from the system can cause a large inhibition of flow. The WSCC Network graph is even more indicative of the relation between edge frequency, critical lines and generators. The inspection of edges that are found in 50% of the bisections will indicate where the largest generators of the network are located.

Figure 7 displays the direct relationship between the frequency of the lines connected to generators and the size of those generators. We performed simple linear regression on the this relationship. The correlation coefficient between the generator size and frequency of edges connecting them is .67. However, when you disregard the two outliers the correlation coefficient is .95. If we perform a hypothesis test it is also found that there is a significant linear relationship between generator size and the frequencies of the lines connecting them to the network. In this case, we therefore conclude that the analysis of frequency to find generators is highly efficient. However, the connectivity of a network can clearly impact the relationship. It should be noted this network has ideal connectivity for this method.

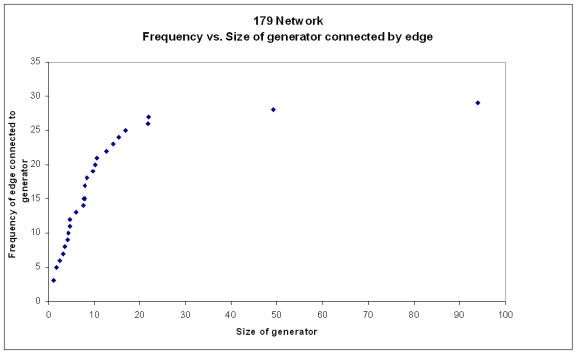

Figure 7: Frequency vs. Size of Generator

4.4 Final Remarks on Frequency Analysis

Because the objective is to minimize cut size and our bisections do not have to be into two connected components, most solutions are combinations of single line bisections. Hence, in the networks tested on, an edge can be initially recognized to be within a direct path to a generator if it is found in 100% of the bisections. In both the 30 and 179 networks the largest generator that is connected by only one edge to the network is within the cutsize of one and is found within 100% of the bisections. In the IEEE 118 bus Network, the edge (8,9) is in a path of two edges which are the sole connection of the second largest generator within the network.

The reason some generators are not partitioned from the graph until higher cutsizes can be understood.

The program begins by removing the least amount of edges that inhibit the flow and partition a given generator. Some generators are of a high degree meaning they have many edges connecting them to the graph. Therefore, a higher cutsize constraint is needed to remove them. However, in most cases it was observed that the larger generators were partitioned from the network by cutting of edges with high frequencies.

A principal relation between frequency and criticality of edges lies within the analysis of the connectivity of the generators to the graph. If an edge is within a path that is crucial to a generator's performance of pushing flow through a network then it is critical. Through the bisection of graphs and the identification of edges with higher frequencies edges within a network that would cut off generators can be found. Going through a network by brute force and removing all combinations of edges can be an arduous task. Our technique can reduce that task significantly. For instance, if edge (i, j) is cut within a bisection and has frequency above some value, it could be considered an edge within a vulnerable area of a network. However, in most cases of larger graphs the frequency list alone will reduce the amount of edges that must be checked within a network. Depending on the weight of importance given to time and error, this method could be sufficient as it has the capability of reducing the time to find critical edges in a network.

5 Linear Optimization for Edge Expendability

In [7], Pinar, Reichert, and Lesieutre attempt to find criticality of power lines in the IEEE 30 Bus and 118 Bus systems by computing the expendability of a line. The less expendable a line is, the more critical it is to a system. Using a parameterization of the continuous version of the problem and solving power flow equations, they compute a relative importance of edges. A similar problem can be solved for our graph theoretical formulation. To compute the criticality of the lines, we formulate the following optimization problem:

$$\begin{array}{ll} \min & ||x|| \\ s.t. & Ax = b \\ & -u \leq x \leq u, \end{array}$$

where A is the node-arc incidence matrix of a graph, b is a vector of supply and demand values of nodes, x is a vector whose values x_i represent the flow going through edge i, and u is a vector of edge capacities. Edges are allowed to have negative flows, as sign represents the direction of power flow. By choosing an appropriate norm to minimize, the function finds a flow that satisfies supply and demand and gives a relative expendability value of edges. The closer to capacity an edge is used, the less expendable it is.

5.1 Results

The optimization problem is solved using Matlab's fmincon function, a constrained nonlinear solver using 2-, 3-, and 4-norms for $\|x\|$. Running time on averages between two and three minutes for the 118-Bus system. It is important to note that the solution method is not optimized for the problem: fmincon is a general nonlinear solver, whereas all of our constraints are linear. In the case of the 2-norm, the problem could be solved more quickly as a linearly constrained least squares problem.

Both the 30 Bus and the 118 Bus systems have especially vulnerable generators. In both graphs, the generators with greatest supply value have small connectivity, while most edges cause an insignificant amount of flow loss when removed from the network. The 2, 3, and 4 norms assign the most flow to the most critical edge. Beyond the edges that cause a significant amount of damage, the solution does not give meaningful results. Edges that cause identical amounts of damage to the network are assigned completely different values. For small graphs, computing criticality of all edges is possible through enumeration of all possibilities. Criticality of each edge is computed by removing the edge from the network and finding the maximum flow on the modified network.

Figure 8 plots the results of this enumeration vs the flow value assigned to each edge by the linear squeeze function using the 4-norm. We see in both that the most significant edge is an outlier, with the rest of the edge having little correlation.

Results for the two other norms on the 118-Bus and results for the 30-Bus , with the most critical line being found and all others having no correlation with their flow assignment x_i.

While this method successfully finds the most critical edge in these networks, we can see mathematically why there is little correlation between criticality of edges and the flow value assigned to them by this optimization problem. The constraints are reasonable for the problem. The supply and

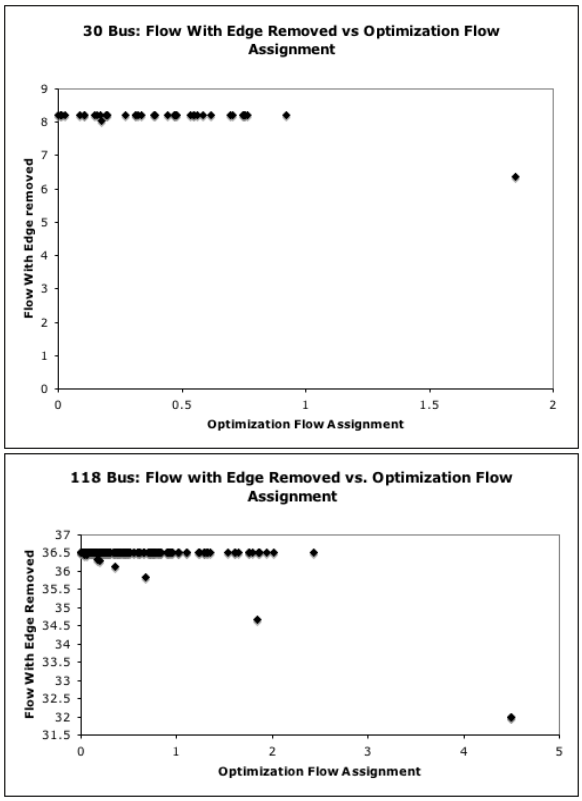

Figure 8: 30-Bus and 118-Bus Matlab Optimization Results Using 4-Norm

demand of the network should be satisfied, and edges cannot carry more energy than their capacity allows. The problem lies in the objective function. In the 30-bus network, the edge that has the most flow is the generator with high capacity and low connectivity. Because its flow travels through a single edge, the x_i that corresponds to that edge must have a flow value equal to the supply value of the generator it connects when minimizing $||x||$. With further research, a similar optimization problem with a more well-defined norm may yield better results for line criticality. It would also be interesting to investigate the relation between line criticalities and multiple line contingencies.

6 Conclusions

This research investigated different methods for determining the criticality of power lines in the electric power grid network. Because of the computational limitations of the network inhibition problem, we considered the inhibiting bisection problem–an effective reformulation. Thus far, it has been proven that the edges cut in a bisection of a network accurately identify the most critical network lines. An in-depth analysis of the most frequently cut edges in network bisections generated these expected results. In every bisection of

each of the three graphs, we conclusively found that the most frequently cut edge was the one connected to the largest generator.

While the inhibiting bisection problem correctly detects the most critical lines in a network, there exist limitations to this approach. Because this problem seeks only bisections of networks, the set of edges cut in each bisection contains some edges that carry little to no importance in terms of criticality. Ultimately, the inhibiting bisection problem generates the intended results; however, it also provides extraneous information that does not provide insight as to which edges are the most critical to the network. Despite this limitation, the inhibiting bisection problem is an effective method for the purposes of this research.

Appendix 1: Frequencies for the IEEE-30 System

Frequencies of each edge.	
(11, 12)	6
(0, 2)	5
(1, 3)	5
(4, 6)	5
(1, 5)	4
(9, 21)	4
(14, 22)	4
(20, 21)	4
(26, 27)	3
(26, 28)	3
(26, 29)	3
(21, 23)	2
(22, 23)	2
(24, 26)	1
(3, 5)	1
(5, 6)	1
(5, 7)	1
(5, 9)	1
(7, 27)	1
(8, 9)	1
(23, 24)	1
(24, 25)	1

Appendix 2: Frequencies for the IEEE-118 System

Frequencies of each edge.

Edge	Frequency	Edge	Frequency	Edge	Frequency
(7, 8)	17	(46, 48)	7	(36, 39)	1
(22, 24)	16	(47, 48)	7	(69, 70)	1
(24, 26)	16	(48, 49)	7	(70, 71)	1
(25, 29)	15	(48, 50)	7	(70, 72)	1
(88, 89)	15	(48, 53)	7		
(88, 91)	15	(98, 99)	6		
(37, 64)	13	(48, 68)	5		
(61, 65)	13	(97, 99)	5		
(65, 66)	13	(99, 102)	4		
(84, 88)	13	(102, 103)	4		
(87, 88)	13	(102, 104)	4		
(46, 68)	12	(102, 109)	4		
(67, 115)	12	(63, 64)	3		
(68, 69)	12	(1, 11)	3		
(68, 74)	12	(2, 11)	3		
(68, 76)	12	(6, 11)	3		
(76, 79)	12	(10, 11)	3		
(78, 79)	12	(11, 13)	3		
(79, 95)	12	(11, 15)	3		
(79, 96)	12	(11, 116)	3		
(58, 60)	11	(64, 67)	2		
(58, 62)	11	(81, 82)	2		
(59, 60)	11	(85, 86)	2		
(79, 98)	10	(59, 61)	1		
(60, 61)	10	(2, 4)	1		
(79, 97)	9	(3, 4)	1		
(109, 110)	9	(4, 5)	1		
(91, 99)	8	(4, 7)	1		
(93, 99)	8	(4, 10)	1		
(99, 100)	8	(7, 29)	1		
(99, 103)	8	(16, 29)	1		
(99, 105)	8	(32, 36)	1		
(48, 65)	7	(33, 36)	1		
(41, 48)	7	(34, 36)	1		

(44, 48)	7	(36, 38)	1

Appendix 3: Frequencies for the WSCC-179 System

Frequencies of each edge.

Edge	Frequency	Edge	Frequency	Edge	Frequency
(74, 75)	29	(17, 21)	1	(81, 168)	1
(77, 78)	28	(17, 23)	1	(81, 170)	1
(32, 33)	27	(25, 137)	1	(82, 83)	1
(28, 29)	26	(29, 30)	1	(102, 132)	1
(116, 117)	25	(31, 32)	1	(102, 133)	1
(138, 139)	24	(35, 36)	1	(106, 131)	1
(63, 64)	23	(35, 54)	1	(106, 133)	1
(13, 14)	22	(35, 62)	1	(106, 172)	1
(147, 148)	21	(36, 46)	1	(106, 174)	1
(6, 7)	20	(37, 44)	1	(106, 176)	1
(3, 9)	19	(37, 46)	1	(109, 118)	1
(42, 43)	18	(37, 57)	1	(109, 171)	1
(10, 11)	17	(37, 58)	1	(111, 112)	1
(142, 143)	17	(40, 48)	1	(113, 123)	1
(145, 146)	15	(40, 55)	1	(113, 125)	1
(156, 157)	15	(40, 56)	1	(113, 126)	1
(34, 83)	14	(46, 47)	1	(113, 128)	1
(66, 68)	13	(49, 50)	1	(117, 121)	1
(110, 111)	12	(49, 51)	1	(117, 127)	1
(3, 4)	11	(50, 61)	1	(117, 129)	1
(136, 137)	10	(51, 61)	1	(117, 130)	1
(15, 16)	9	(53, 54)	1	(117, 132)	1
(0, 2)	8	(61, 62)	1	(137, 145)	1
(100, 101)	7	(62, 137)	1	(140, 145)	1
(114, 115)	6	(62, 140)	1	(140, 151)	1
(159, 160)	5	(67, 70)	1	(141, 144)	1
(40, 41)	3	(69, 70)	1	(143, 144)	1
(37, 38)	2	(72, 76)	1	(151, 152)	1
(45, 46)	2	(80, 85)	1	(161, 162)	1
(0, 1)	1	(80, 89)	1		
(1, 6)	1	(80, 93)	1		
(5, 6)	1	(81, 87)	1		

(6, 161)	1	(81, 92)	1
(10, 18)	1	(81, 96)	1
(10, 20)	1	(81, 166)	1

Appendix 4: Visualization of Networks

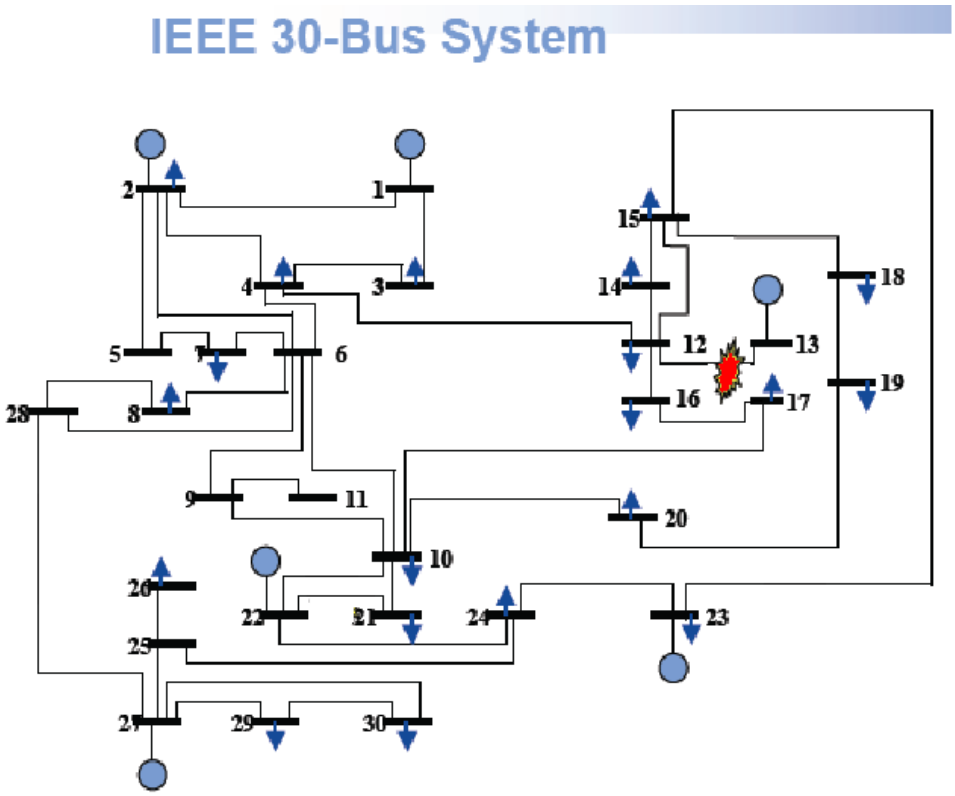

Figure 9: IEEE 30 System

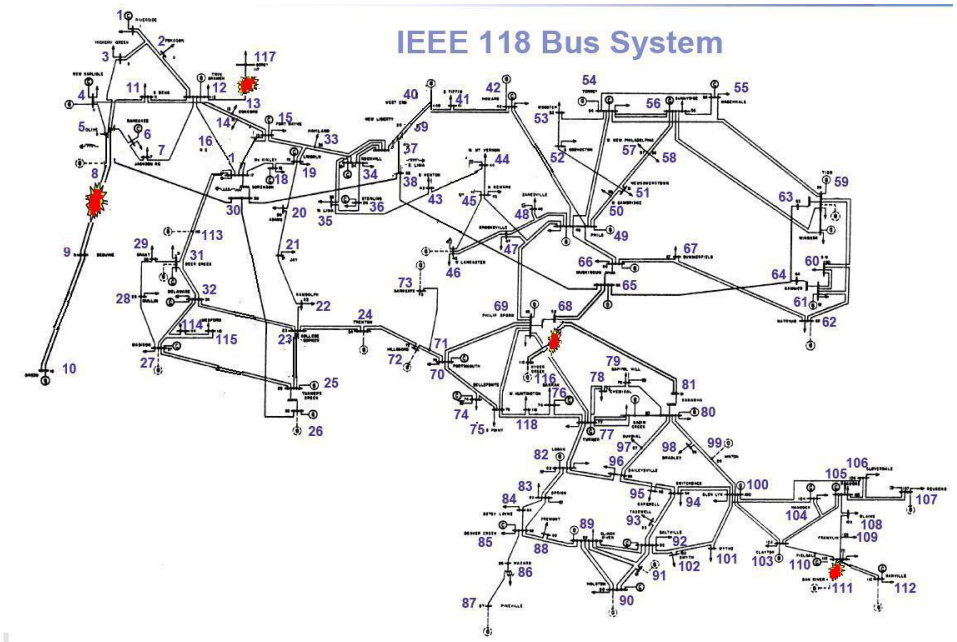

Figure 10: IEEE 118 System

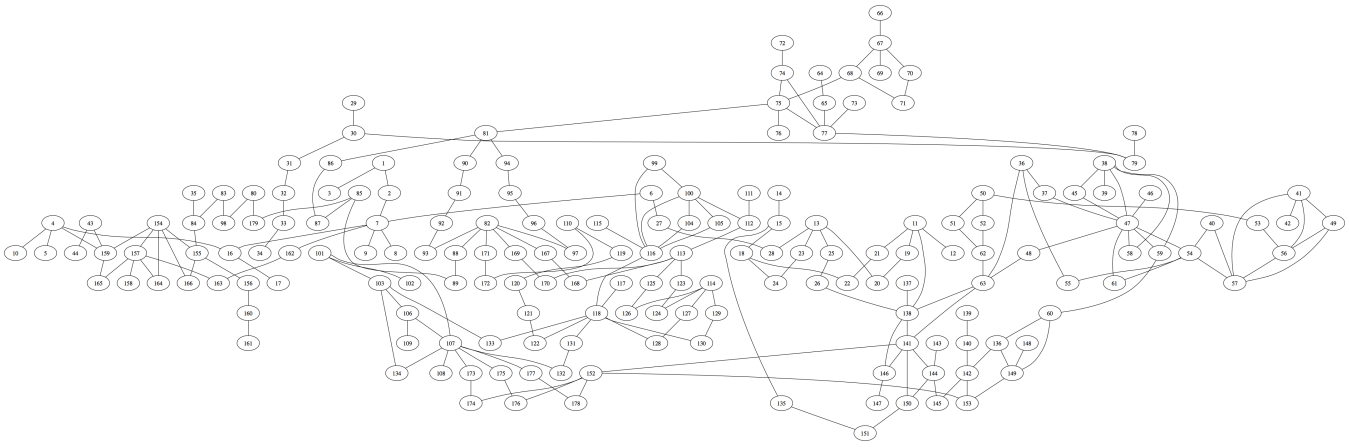

Figure 11: WSCC 179 System

7 Acknowledgements

We would like to thank our director, Dr. Ricardo Cortez, and graduate assistant Edgar Lobaton.
This work was supported by National Security Agency (NSA) grant H98230-07-1-0084 and also received funding from the host institution The Mathematical Sciences Research Institute (MSRI). Ali Pinar and Juan Meza was supported by the Director, Office of Science, Division of Mathematical, Information, and Computational Sciences of U.S. Department of Energy under contract DE-AC03-76SF00098.

References

[1] T. Aura, M. Bishop, and D. Sniegowski. Analyzing Single-Server Network Inhibition. *Computer Security Foundations Workshop*, 2000.

[2] Yuri Boykov and Vladimir Kolmogorov. An Experimental Comparison of Min-Cut/Max-Flow Algorithms for Energy Minimization in Vision. In *Energy Minimization Methods in Computer Vision and Pattern Recognition*, pages 359–374, 2001.

[3] Canada-U.S. Power System Outage Task Force. Interim Report: Causes of the August 14th Blackout in the United States and Canada. Technical report, 2003.

[4] M.T. Heath. *Scientific computing: an introductory survey.* McGraw-Hill, 2002.

[5] Cynthia A. Phillips. The Network Inhibition Problem. In *STOC '93: Proceedings of the twenty-fifth annual ACM symposium on Theory of computing*, pages 776–785, New York, NY, USA, 1993. ACM Press.

[6] A. Pinar, Y. Fogel, and B. Lesieutre. The Inhibiting Bisection Problem. Submitted to ACM 19th Symposium Parallel Algorithms and Architectures(SPAA) 2007.

[7] A. Pinar, A. Reichert, and B. Lesieutre. Computing Criticality of Lines in Power Systems. *IEEE International Symposium on Circuits and Systems*, 2007.