RICE UNIVERSITY

Fort Neighborhoods: A Set Cover Formulation for Power Domination in Graphs

by

Logan Smith

A THESIS SUBMITTED
IN PARTIAL FULFILLMENT OF THE
REQUIREMENTS FOR THE DEGREE

Master of Arts

APPROVED, THESIS COMMITTEE:

Illya V. Hicks, Chair
Professor of Computational and Applied Mathematics

Andrew J. Schaefer
Noah Harding Chair and Professor of Computational and Applied Mathematics

Moshe Y. Vardi
Karen Ostrum George Distinguished Service Professor in Computational Engineering

Houston, Texas

May, 2018

ABSTRACT

Fort Neighborhoods: A Set Cover Formulation for Power Domination in Graphs

by

Logan Smith

This thesis introduces a novel separation algorithm for calculating power domination numbers and minimum power dominating sets in graphs. Additionally, it shows how the existence of solutions of special forms can be exploited by computational methods. Power domination studies arise from a key problem in electrical engineering concerning placements of monitoring devices known as Phasor Measurement Units (PMUs). PMUs are installed in electrical networks for early detection of electrical imbalances, enabling corrective actions to mitigate outages. In graph representations of electrical networks, power dominating sets denote locations at which PMUs can be placed to monitor entire networks. Due to PMU installation costs of up to $200,000, it is desirable to identify minimum power dominating sets and their cardinality, known is the graph's power domination number. Unlike prior methods, this work exploits crucial yet previously neglected graph structures: the neighborhoods of zero forcing forts. Utilizing these structures, algorithms are devised for calculating minimum power dominating sets and the power domination numbers of graphs. Computational experiments demonstrating an order of magnitude improvement over previous methods are presented.

Acknowledgments

I would like to thank all of the mentors and teachers who have helped me get to where I am today. Without the support and guidance provided by each and every one of you, I would surely not have made it this far.

Specifically, I would like to thank my advisor Dr. Illya Hicks for his direction in my academic pursuits and the other members of my Master's Thesis Committee: Dr. Andrew Schaefer and Dr. Moshe Vardi.

Finally, I would like to thank my friends and family for their constant patience and loving support for all of my endeavors.

Contents

Chapter 1

Introduction

This thesis proposes new algorithms for the computation of power domination numbers and minimum power dominating sets in graphs. While several approaches have been previously published in power domination literature, direct comparison reveals that the proposed algorithms exhibit significantly improved runtime performance. The proposed algorithms depart from earlier approaches by exploiting the presence of zero forcing fort neighborhoods, a crucial graph structure that has been neglected in previous methods. It is shown here that power dominating sets must intersect each of these neighborhood sets. Additionally, this thesis investigates the properties of zero forcing fort neighborhoods and gives necessary and sufficient conditions for a set of vertices to be the neighborhood of a zero forcing fort. These conditions, paired with a novel vertex partitioning scheme, facilitate practical algorithms for the identification of minimum power dominating sets.

In the remainder of this chapter, preliminary notation, the power domination graph color changing process, and the zero forcing graph color changing process are introduced. Then, zero forcing forts are defined and their relation to the study of zero forcing is discussed. Next, existing computational methods for zero forcing and power domination are reviewed. Finally, the introduction concludes with an outline of the remaining chapters of the thesis.

1.1 Preliminary Graph Theoretical Notation

The following notation will be used throughout this thesis. A *graph*, $G = (V(G), E(G))$ is an ordered pair made up of a set of *vertices* $V(G)$, and a set of *edges*, $E(G) \subseteq V(G) \times V(G)$. Often these sets can be simply abbreviated as V, E, and the graph parameters *order*, $n = |V|$, and *size*, $m = |E|$, are used for convenience. Two vertices u, v in V are said to be *adjacent* if the edge $uv = e$ is in E. Additionally, $e = uv$ is said to be incident on its end vertices, $u, v \in V$. The degree of a vertex v is the number of edges incident on v. The *open neighborhood* of a vertex v is denoted $N(v)$ and is defined as the set of vertices adjacent to v, or more formally: $\{u \in V : uv \in E(G)\}$. The *closed neighborhood* of a vertex is denoted $N[v]$, and is defined $N[v] := N(v) \bigcup \{v\}$. The open neighborhood of a set S is then defined as $N(s) := (\bigcup_{v \in S} N(v)) \backslash S$. Similarly, the closed neighborhood of a set S, is defined as $N[S] = \bigcup_{v \in S} N[v]$. The degree of a vertex is denoted $\deg(v)$, and is the equal to the number of edges incident on v. A graph $H = (V_H, E_H)$ is said to be a *subgraph* of G if $V_H \subseteq V$ and $E_H \subseteq E$. A subgraph H can be *induced* in G with a vertex set S, denoted $H = G[S]$, by considering the vertices in S and the edges with both end vertices in S.

An edge xy is a *loop*, if $x = y$. Two edges xy, uv are *parallel* if $x = u$ and $y = v$ or $x = v$ and $u = y$. A graph with no loops or parallel edges is called a *simple* graph, and all graphs in this thesis are assumed to be simple. A *digraph* is a graph with directed edges, often referred to as *arcs*. Here, edges are thought of as ordered pairs of vertices instead of simply as pairs of vertices. Each arc has one end vertex designated as its *tail* and one end vertex designated as its *head*. Arcs are said to be incident on their head vertices, and the definitions of neighborhoods and degrees of vertices in digraphs can be defined analogously to their undirected versions. A *path* in G is a sequence of vertices $(v_1, v_2, \ldots, v_k)$ such that for $i = 1, 2, \ldots, k-1, v_i$ is adjacent to v_{i+1}. G is called *connected* if for any two vertices u, v, there is a path from u to v. A graph is called a cycle if it is connected and all vertices

have degree 2. A graph is called a tree if it is connected and does not contain any cycles. A component of a graph is a maximal set of vertices that induce a connected subgraph.

Additional notation and well known properties of graphs are provided by Bondy and Murty [7].

1.2 Power Domination

The graph theoretical power domination problem is equivalent to a key sensor placement problem in electrical engineering known as the PMU Placement Problem. In electrical networks, sensors called Phasor Measurement Units (PMUs) can be installed at power substations to measure the currents and phase angles of transmission lines incident upon the substations. By measuring the states of these variables, in all areas of the network, incipient electrical imbalances can be detected and mitigated. Early detection of these imbalances enables corrective measures to prevent power outages and potential damage to energy infrastructure. While it would be possible to fully observe electrical networks by placing PMUs at all power substations, the cost of the sensors makes this approach impractical. Instead, PMUs can be placed at a subset of the substations, allowing a portion of the network to be directly observed. Then, by using circuit laws such as Kirchhoff's Voltage Law, the states of additional areas within the network may be inferred and observed indirectly. Many sensor placement strategies seek to either directly or indirectly observe the entire network, while employing a minimum number of PMUs. This question, of how best to place PMUs in general electrical networks, is referred to as the PMU Placement Problem. Within electrical engineering literature several approaches and variations to the PMU Placement problem have been well studied, such as placement in stages [42] and placement under probabilistic equipment failures [43]. Manousakis, Korres, and Georgilakis [39] provide a taxonomic survey of placement strategies.

Due to the complex chain of inferences needed to determine which substations and transmission lines can be observed indirectly, many approaches for the PMU Placement Problem rely upon heuristics or probabilistic placement strategies. However, this chain of inferences can be simply modeled by abstracting the PMU placement problem into a graph problem. Electrical networks can be thought of as graphs, where vertices represent power substations and edges represent transmission lines. In a graph $G = (V, E)$ a set of vertices $S \subseteq V$ is designated to have sensors, and the vertices that are colored by the following graph color changing rule are either directly or indirectly observed by the sensors in S.

Power Domination Color Changing Rule: [5]

1. Color all vertices in or adjacent to S.
2. While there is a colored vertex adjacent to exactly one uncolored vertex u, color u.

The set of vertices which can be colored from an initial set of vertices S is called the *power domination closure* of S, and is denoted as $cl_P(S)$. If $cl_P(S) = V(G)$, or equivalently if all vertices in a graph G can be colored by this process, then S is said to be a power dominating set in G. The cardinality of a minimum power dominating set in G, denoted as $\gamma_P(G)$, is called the *power domination number* of G. In the example of electrical networks, if the locations where sensors are placed correspond to a power dominating set in the network's graph representation, then the sensors are able to fully observe the electrical network; $\gamma_p(G)$ is the minimum number of sensors needed to fully observe the electrical network. The decision version of the *power dominating set problem* in graphs is the following: Given a graph G, is there a power dominating set of cardinality k or less?

Haynes, Hedetniemi, Hedetniemi, and Henning [33] first established a graph color changing rule to describe the observability of power substations, based on PMU placements. Since

PMUs are able to directly observe the transmission lines incident to their locations, it was noted that the set of observable locations resembles the well known dominating set problem in graphs. In their publication, Haynes et al. also show that the power domination problem is NP-Complete, even in the special cases of bipartite and chordal graphs. It was later further shown by Brueni and Heath [11] that the power domination problem is also NP-Complete in the case of planar bipartite graphs. Additionally, the color changing rule provided by Haynes et al. was later simplified by Benson [5] to the graph color changing rule shown above. Since then, theoretical and computational studies have been done to bound the sizes of solutions [46], and to provide polynomial complexity algorithms for finding solutions in certain graph families [15].

Variants of the power domination problem have also been studied in previous literature, often motivated by other concerns within electrical engineering. One such example is the connected power domination problem [10], in which the initial set S is also required to be connected. This problem is motivated by the need for additional equipment to process the information collected by PMUs. To lower the expenses of supporting infrastructure, PMUs can be placed in compact regions so that the number of required processing centers is also minimized. Another problem variant is the restricted power domination problem proposed by Bozeman, Brimkov, Erickson, Ferrero, Flagg, and Hogben [8]. In this problem, solutions are required to include a set of specified vertices. This corresponds to the incremental nature in which power grids are constructed and adapted over time. In early iterations of the electrical network, sensors may be incorporated into the power grid. As the network is then modified, these sensors may remain operational and can be utilized without the cost of repurchase or reinstallation. Thus, requiring the use of these sensors when considering the placements of additional sensors allows for the placements of additional sensors to be optimized in subsequent iterations of the power grid.

A wealth of similar graph theoretic problems motivated by practical concerns in PMU placement still remain, and presents an interesting area of future research. In current PMU placement literature, the effects of equipment failure [40, 43, 30] and malicious data injections [38, 35, 22] are highly active, and could motivate such future works.

1.3 Zero Forcing

Zero forcing is a phenomenon that was independently discovered by mathematicians working to establish bounds for the minimum rank of combinatorial matrices [44] and by physicists seeking to control complex quantum systems while only directly affecting a small quantum subsystem [12]. Since then, zero forcing has been well studied in mathematical literature, including notably: characterizations of how the zero forcing number of a graphs change with respect to graph operations [25], zero forcing in special families of graphs [26], and bounds on the zero forcing number of graphs [2, 29].

Similar but distinct from the power domination problem, the zero forcing graph process can also be described by a set of vertices and a color changing rule. Zero forcing is also a widely studied graph coloring process, and has its own applications and techniques. It is introduced here because the techniques developed for its study can be adapted for power domination. Formally, given an initial set of colored vertices in G, the zero forcing color changing rule is stated as follows:

Zero Forcing Color Changing Rule:

1. While there is a colored vertex adjacent to exactly one uncolored vertex u, color u.

The *zero forcing closure* of a vertex set $S \subseteq V(G)$, denoted $cl_Z(S)$, is defined as the set

of vertices that are colored after the zero forcing color changing rule is applied until no more vertices can be colored. If $cl_Z(S) = V(G)$, then S is called a a *zero forcing set* in G. Additionally, the cardinality of the minimum zero forcing sets in a graph, denoted $Z(G)$, is the *zero forcing number* of G. If a vertex v is colored and had a colored neighboring vertex u such that v was its only adjacent neighbor, u is said to *force* v. Following a sequence of forces, a vertex u may be able to force a vertex, despite not being part of the original set of colored vertices. The orders in which vertices are forced can be recorded to describing possible ways in which the color changing rule can be applied. These sequences are known as *forcing chains* and were introduced in the context of undirected graphs by Barioli, Barrett, Fallat, Hall, Hogben, Shader, Driessche, and Holst. [4]. Forcing chains are often employed as logical tools for analyzing the properties of zero forcing sets. The following proposition makes use of this concept:

Proposition 1.1. If S_1 is zero forcing in G, $S_1 \subseteq S_2 \subseteq V(G)$, then S_2 is zero forcing in G.

Proof. If S_1 is zero forcing, then there exists a forcing chain F in which all vertices in $V(G)\backslash S_1$ are forced, with initial set of colored vertices S_1. Modify F by removing each force in F in which a member of the set $(V(G)\backslash S_1)\backslash S_2$ is forced to obtain F'. Then for any force in F', consider initial set of colored vertices S_2 and without loss of generality assume vertex v forces vertex u. Since this force is valid in F, at this point in the forcing sequence each neighbor of v, with the exception of u is colored. In the forcing chain F' originating from vertex set S_2 each neighbor of v, with the exception of u, is in S_2 or colored at a previous step. Since all vertices in $V(G)\backslash S_2$ are colored by F', S_2 is zero forcing in G. □

A related concept used for analyzing of zero forcing sets is the propagation time of a zero forcing sets. This parameter describes the maximum number of forces needed to force

any vertex in a graph, with respect to a particular zero forcing set. Beyond a theoretical curiosity, in applications this parameter relates to how tightly systems are controlled, and was originally discovered in quantum control theory [41, 13]. Later introduced by Hogben, Kingsley, Meyer, Walker, and Young [34], the minimum propagation time of a graph, which is denoted $pt(G)$, is the minimum number of forces needed for any vertex in a graph to be forced for any minimum zero forcing set in G. In addition to stronger results for special families of graphs, the following proposition holds for general graphs:

Proposition 1.2. For any graph G, the propagation time $pt(G) \leq n-1$.

First acknowledged by Dean, Ilic, Ramirez, Shen, and Tian [21], since the zero forcing color changing rule and the second line of the power domination color changing rule are identical, the following proposition holds:

Proposition 1.3. [21] $S \subseteq V$ is a power dominating set in G if and only if $N[S]$ is a zero forcing set in G.

This proposition offers an immediate relationship between zero forcing and power domination. This correspondence allows for both theoretical and computational results in zero forcing to be adapted and applied to the study of power domination.

Originally introduced in the PhD Thesis of Caleb Fast [28], subgraphs known as *forts* can be used to enhance the study of zero forcing. Intuitively, forts can be thought of as sets of vertices that cannot be colored by applying the zero forcing color changing rule, if all members of the fort are all initially uncolored. Formally, forts are defined as follow:

Definition 1.1. A non-empty set of vertices $F \subseteq V(G)$ forms a *fort* in G if for any $v \in V(G) \backslash F$, $N(v) \bigcap F = \varnothing$ or $N(v) \bigcap F \geq 2$.

As an immediate consequence of the zero forcing color changing rule, since each vertex that is adjacent to the fort is adjacent to at least 2 members of the fort even if all vertices

outside of a fort are colored the vertices inside of the fort cannot be forced. Any fort with any members that are colored during the zero forcing color changing process must have began the process with at least one colored vertex. The following theorem generalizes this idea and is presented by Fast as Theorem 5.2 [28].

Theorem 1.1. [28] If S is a zero forcing set in G and F is a fort in G, then there exists a vertex $v \in S \cap F$.

The set covering approach for computing zero forcing numbers in graphs, discussed in Section 1.4, is motivated by this theorem. It provides a necessary condition that must be satisfied by any zero forcing set. If a set S does not contain a specific fort F, then S can be augmented by adding a member of F. While not directly stated in this theorem, finding a set that contains a member of all forts in a graph ensures that all uncolored vertices can be colored by the zero forcing process. This concept is the intuition behind the set covering approach discussed in the next section.

As a final note on zero forcing, beyond the standard zero forcing problem, several similar coloring processes and generalizations of zero forcing have also been studied. These problems arise in various applications that can be described with slightly altered color changing rules. Specifically concerned with infectious disease spread, Dreyer and Roberts [24] introduced a coloring process in which vertices are colored if they have at least k colored neighboring vertices. A direct generalization of zero forcing is also provided by Amos, Caro, Davila, and Pepper [2] in which an uncolored vertex can only be colored if it has k colored neighbors for which it is the only uncolored neighbor. Furthermore, the restricted zero forcing problem analogous to the restricted power domination problem was also introduced by Bozeman et al. [8]. Related color changing processes such as these are motivated by other applications, but the ideas presented in this thesis are likely to have parallels in these related paradigms.

1.4 Computational Methods in Zero Forcing

Previous work regarding the computation of zero forcing sets has been done through the use of Integer Linear Program (ILP) models and exponential complexity algorithms. Discussed in detail in Section 1.5, one flavor of the ILP models developed for the zero forcing problem has been modified for use in power domination computational studies.

This ILP model is known as the infection model because it models the way colorings iteratively spread throughout the graph, akin to the creeping spread of an infection. Expanding upon this notion, the infection model works by considering the possible chronological chains of forces originating from an initial set of colored vertices. Solutions for this model represent a zero forcing set and a specific chronological chain of forces such that all vertices in the graph are colored. In its objective function, the model minimizes the cardinality of the set of initially colored vertices, and thus optimal solutions correspond to minimum power dominating sets. Intuitively, the solutions for this model offer an optimal solution for the zero forcing set problem, as well as a certificate of its feasibility. Constraints are then introduced to verify that the candidate zero forcing set is indeed a valid zero forcing set.

Formally, the model works by considering a constructed directed graph. Given a graph $G = (V, E)$, a digraph D is obtained by replacing each edge $e = xy$ in E with an arc xy and yx. The model also introduces S, a binary variable over V indicating the members of a zero forcing set, X, an integer variable over V indicating the timestep or wave in which each vertex is forced, and a binary variable y over the arcs in D indicating which vertices are responsible for forcing other vertices.

Stated below is the zero forcing infection model formulated by Brimkov, Fast, and Hicks [9], appearing as Integer Program 1.1 :

Integer Program 1.1. Zero Forcing Infection Model [9]:

$$\begin{aligned}
\text{Min:}\quad & \textstyle\sum_{v\in V} s_v \\
\text{s.t.}\quad & s_v + \textstyle\sum_{e\to v} y_e = 1 && \forall v \in V \quad (1)\\
& x_u - x_v + (T+1)y_e \le T && \forall e=(u,v)\in E \quad (2)\\
& x_w - x_v + (T+1)y_e \le T && \forall e=(u,v)\in E, \forall w \in N(u)\backslash\{v\} \quad (3)\\
& x_v \in \{0,1,\ldots,T\}, \qquad s_v, y_e \in \{0,1\} && \forall v \in V, e \in E
\end{aligned}$$

In this model, constraint (1) ensures that each vertex is either contained the initial set of vertices, or is forced by an adjacent vertex exactly once. Constraints (2) and (3) then make certain that all forcings are valid by the color changing rule. Constraint (2) requires that if the variable y_e is non zero, then the foot of arc e was colored at a time step predating the timestep that the head of e was colored. Constraint (3) guarantees that the foot of the arc has exactly one uncolored neighbor at the time the forcing occurs at.

This model also includes an integer parameter T, which is a tunable parameter restricting the total number of forcing waves solutions are allowed to have. Thus, beyond finding minimum zero forcing sets, this model can also be used to find minimum zero forcing sets subject to a maximum propagation time. Note that by Proposition 1.2, if $T \geq n-1$ the characteristic vector of any zero forcing set in the original graph with its corresponding forcing edges and timestep information is a feasible solution to 1.1. The correctness of this model was verified by Brimkov, Fast, and Hicks [9].

Besides the infection model, an alternative approach is introduced by Fast [28] in his PhD thesis. From Theorem 1.1, it is clear that any zero forcing set must include a member of each fort. This idea can be exploited to create a set covering problem and corresponding integer program. In this approach, the vertex sets of forts form sets that must be covered by a zero forcing set. Thus, the following integer program finds a minimum cardinality set

which covers all forts for a graph G.

Integer Program 1.2. Zero Forcing Set Cover IP [28]

$$\begin{aligned} \text{Minimize} \quad & \textstyle\sum v \in V s_v \\ \text{Subject To} \quad & \sum_{v \in F} s_v \geq 1 \qquad \forall F \in \mathscr{F} \\ & s_v \in \{0,1\} \end{aligned}$$

In this model, $\mathscr{F}$ is the set of vertex sets of forts in G. Unfortunately, some graphs may contain an exponential number of distinct vertex sets that are forts. One such example is the k-star graph, the graph obtained by adding k leaves to a vertex. In this graph any set of two or more leaves forms a fort, thus there are 2^{k-2} distinct forts in a k-star.

However, the Zero Forcing Set Cover integer program is a classical set cover problem which has been well studied in combinatorial optimization literature. As such, well known results including the necessary and sufficient conditions for set cover constraints to be facets, provided by Balas and Ng [3], can be considered. In order to reproduce these results, additional notation must be introduced. Let $P := conv\{x \in \{0,1\}^n : Ax \geq 1\}$ be the set cover polytope, with rows $(A_1, \ldots, A_n)$ of A indicating sets in a constraint set M.

Theorem 1.2. [3] The inequality $\sum_{j \in A_i} x_j \geq 1$ is facet inducing for P if and only if

1. There exists no constraint set S_k in M such that the constraint set S_j contains S_k.

2. For each item not included in the constraint set S_j, there exists an item k' in S_j such that for any constraint set $S \in M$ that includes an item k such that $S \subset S_j \cup k'$, then k' is also in S.

In the case of the Zero Forcing Set Cover Polytope, the first statement of Theorem 1.2 has an intuitive meaning: facet defining constraints for the Zero Forcing Set Cover IP must be minimal forts. In practicality, this means that constraints corresponding to non-minimal

forts can be excluded from the Zero Forcing Set Cover Polytope. In an obvious sense, if a set contains a smaller set which is covered by a potential solution to the set cover problem, the larger set is also covered. Thus, this theorem ensures that only constraints corresponding to minimal forts must be included in the integer program formulation.

To solve this integer program, an algorithm for finding forts is also required. To this end, Fast [28] proposes several integer program models that can be used to identify forts. These models find forts that are minimum with respect to a set of vertex weights c, and exclude certain forbidden vertices in a set $S \subseteq V(G)$. The results presented in Fast's thesis also include more complex integer program models which are designed to find facet-inducing fort inequalities. This goal is accomplished by ensuring that the forts that are found satisfy the second property of Theorem 1.2. For simplicity, only the most basic of Fast's integer program models is reproduced below:

Integer Program 1.3. Fort Finding IP [28]

$$\begin{aligned}
\text{Minimize} \quad & \sum_{v \in V} c_v x_v & & \\
\text{Subject To} \quad & \sum_{v \in V} x_v \geq 1 & & (1) \\
& x_w - x_v + \sum_{i \in N(w) \setminus \{v\}} x_i \geq 0 & \forall vw \in E \quad & (2) \\
& x_v \in \{0,1\}, & \forall v \in V &
\end{aligned}$$

In this model, x_v variables indicate the vertices that are potential members of a discovered fort. Additionally, Fast [28] shows that optimal solutions for this integer program correspond to minimum weight forts. Constraint (1) ensures solutions are non-empty, while constraint (2) requires each vertex u with $x_u = 0$ adjacent to an vertex u indicated to be in the fort is adjacent to at least one other member of the fort. In practice, this model can be used to generate the constraint set for the Zero Forcing Set Cover ILP. As previously stated however, the number of constraints may be large relative to the problem size.

Ultimately, this problem with the use of a separation algorithm. Similar to the historic Traveling Salesman Problem studied by Dantzig, Fulkerson, and Johnson [20], this is done by formulating a solution that satisfies a subset of the constraints for a master problem, and then incorporating constraints that the interim solution violates. This process of finding partial solutions and violated constraints is then performed repeatedly until a solution is found and verified to satisfy all problem constraints. This commonly used scheme often allows for solutions to be found that satisfy large systems of constraints, without explicitly finding the entire set of constraints that must be fulfilled [32, 31, 19]. Finding violated constraints can thought of as a series of subproblems that must be solved while ultimately progressing towards the goal of solving a master problem. The computational complexity of solving these subproblems is also a matter of concern. This ensures that the methods used are efficient as possible, and that exponential complexity algorithms are not used to solve problems that could instead be solved by more tractable strategies.

In practice, separation algorithms are often computationally efficient methods. As such it is expected that an efficient separation algorithm should outperform the infection model. In the case of Zero Forcing and Power Domination, this benefit in performance is demonstrated by Brimkov, Fast, and Hicks [9], as well as by the work in this thesis. It can be noted however, that this improvement in performance is dependant on the graphs in which the techniques are compared. In some types of graphs, such as small world networks and social networks, the infection model and the separation algorithm has similar performance results. In the graphs corresponding to electrical networks however, the separation algorithm can be seen to outperform the infection model. These graphs are often more sparse and may contain a smaller number of forts, requiring fewer subproblems to be resolved by a separation algorithm.

However, there is another algorithm in zero forcing literature which can be observed

to outperform both of the previously introduced approaches. This approach is known as the Wavefront Algorithm, and was introduced by Butler, DeLoss, Grout, Hall, LaGrange, McKay, Smith, and Tims [14], with code available for download. The performance of the wavefront algorithm was evaluated by Fast [28] and compared with the other methods introduced in this section. In the case of some types of graphs, the wavefront algorithm greatly outperforms both the set cover approach and the infection model approach, while in others its performance is comparable. Brimkov, Fast, and Hicks also consider the wavefront algorithm, and prove its correctness.

1.5 Computational Methods in Power Domination

Aazami [1] introduces the first integer program formulation for the power dominating set problem. This formulation is similar to the Zero Forcing Infection Model in that it requires the use of a parameter T, limiting the number of rounds the color changing rule can be applied. As with Zero Forcing Infection Model, any power dominating set fully colors V in fewer than $n-1$ color changing applications. Thus, T can be chosen to be large enough so that solutions can always be found.

Several years later, Fan and Watson[27] introduced a similar integer program formulation for a power domination like problem. In their work, they consider optimal sensor placement in electrical grids, but with the presence of zero injection buses. Zero injection buses are electrical buses which are known to not affect the voltage levels of adjoining buses. However, the information imparted by zero injection buses leads to a different problem formulation than the standard graph theoretic power dominating set problem. Thus, the method presented by Fan and Watson is not immediately comparable to the others presented in this thesis. For completeness however, this model is acknowledged.

Finally, an additional integer program formulation for power domination was also intro-

duced by Brimkov, Mikesell, and Smith [10]. This formulation also includes a parameter T limiting the total number of color changing rounds. In their study, Brimkov et al. demonstrate how runtime performance varies significantly with this parameter T, and find no clear choice for T which offers consistently improved performance. Nonetheless the model they propose requires a reduced number of variables and outperforms the model introduced by Aazami in their presented computational experiments. As with Integer Program 1.1, when considering a graph G the full algorithm constructs a digraph by replacing the edges of G with arcs facing both directions. The following model is then solved:

Integer Program 1.4. Power Domination Infection Model [10]:

$$\begin{aligned}
\text{Min:} \quad & \textstyle\sum_{v\in V} s_v & & \\
\text{s.t.} \quad & s_v + \textstyle\sum_{e\to v} y_e = 1 & \forall v \in V \quad & (1)\\
& x_u - x_v + (T+1)y_e \leq T & \forall e=(u,v)\in E \quad & (2)\\
& x_w - x_v + (T+1)y_e \leq T + (T+1)s_u & \forall e=(u,v)\in E, \forall w \in N(u)\backslash\{v\} \quad & (3)\\
& x_v \in \{0,1,\ldots,T\}, & \forall v \in V, & \\
& s_v, y_e \in \{0,1\} & \forall e \in E &
\end{aligned}$$

Also proven by Brimkov, Mikesell, and Smith [10], optimal solutions for the ILP correspond to minimum power dominating sets in G. In the computational experiments performed in Chapter 5, I reimplement this algorithm and use it as a point of comparison for the methods introduced in this thesis. When applicable, improvements made for these novel approaches are also applied to this method, and demonstrate consistently improved performance for this model as well.

While integer program modelss for the power dominating set problem have been introduced in three separate publications, all three models take generally similar approaches. In essence, each model has variables corresponding to an initially colored vertex set, possible edges along which colorings occur, and the times at which vertices are colored. Each

model then features a system of constraints verifying that the directions and times that these colorings occur at are valid under the power domination color changing rule. This thesis works to introduce a method distinct from these previous infection style approaches. Instead, the methods introduced here more closely resemble the set cover problem reduction for zero forcing used by Fast [28]. Ultimately, this style of model can be effectively solved through the use of a separation algorithm, unlike the infection model. This strategy improves the performance of these novel methods, giving them a strong advantage over the current approaches seen in the power domination literature.

1.6 Thesis Outline

The primary focus of this thesis is to create a separation algorithm similar to the algorithm described by Fast [28] in his thesis, but instead for the power dominating set problem. As was the case for computational approaches in zero forcing, it expected that this approach will outperform the infection model approaches, which are the current state of the art in power domination literature.

In order to accomplish this alternative approach for power domination, a new set cover problem is defined where the neighborhoods of forts correspond to constraints. This approach requires an analysis of fort neighborhoods, a graph structure that has not been explored in previous literature. Additionally, a practical means of identifying these new graph structures is required for a runtime efficient method. To improve the runtime efficiency of this method, the properties of typical electrical networks is exploited. Using this intuition, I verify the existence of optimal solutions with a special form. Then, I show how the structure of these solutions can lead to a dimensional reduction of the power dominating set problem, further improving the performance of the proposed separation algorithm, as well as improving the existing approaches for the power dominating set problem.

The remaining contents of this thesis are aligned as follows. Chapter 2 provides a novel vertex partitioning scheme useful for analyzing the properties of power dominating sets, forts, and neighborhoods of forts. This chapter also gives a proof guaranteeing the existence of minimum power dominating sets made up entirely of degree three or more vertices in nearly all graphs. The theoretical results and lemmas given in this chapter create a framework which is exploited throughout the later chapters.

Chapter 3 uses the findings of Chapter 2 to characterize the forms of fort neighborhoods in the neighborhoods of high degree vertices. Areas in graphs where high degree vertices are adjacent to low degree vertices form critical regions which can provide insight on the detection of fort neighborhoods. With this intuition, necessary and sufficient conditions for vertex sets to be fort neighborhoods are then derived. These conditions are used to create an ILP for finding minimum fort neighborhoods. The correctness of this ILP is verified and used as an approach for solving subproblems in the power domination separation algorithm in the following chapter. Additionally, the complexity of finding minimal weight fort neighborhoods is explored, and a practical algorithm and random heuristic for finding these sets are given. These algorithms act as subprocesses for the separation algorithm introduced in the next chapter.

Chapter 4 continues upon the work in Chapter 3 by using the integer program defined in Chapter 3 with a set cover formulation for solving the power dominating set problem. A separation algorithm for this approach is given and proven to correctly identify minimum power dominating sets. Variations of the separation algorithm are discussed and compared and details of their implementation are considered.

Chapter 5 compares the performance of the separation algorithm presented in Chapter 4 with the performance of the previous infection model presented by Brimkov, Mikesell, and Smith [10]. These experiments demonstrate that the the proposed algorithms have

significantly improved performance over previous methods. Furthermore, it is shown that the variable elimination method introduced in Chapter 2 provides a performance improvement for both the separation algorithm and the previously introduced infection model.

Finally, Chapter 6 provides a summary of results, and presents several open problems that are promising avenues for future research efforts.

Chapter 2

The Junction Vertex Partition

Since the power domination problem is motivated by an optimization problem in electrical networks, it is reasonable to consider the typical structures present in graphs corresponding to electrical networks. The graphs present in standard IEEE data sets often have a subset of high degree vertices and contain many chains of degree one and two vertices that form paths adjacent to one or more of the high degree vertices. An example of one such commonly used graph is presented as Figure 2.1.

The main focus of this chapter is to introduce a graph vertex partition which will be used in subsequent chapters to analyze the properties of power dominating sets. In this chapter, I also present a proof guaranteeing the existence of minimum power dominating sets comprised solely of degree three or higher vertices, in all connected graphs except for cycles and paths. A similar statement is included by Haynes et al. [33], however I provide a counterexample to that statement and give the additional assumptions needed for the statement to be made correct.

Due to the weakness of the necessary assumptions, this special class of optimal solution exists in virtually all graphs corresponding to electrical networks. Additionally, the existence of these special minimum power dominating sets allow several computational approaches presented in later sections to consider only a restricted subset of possible solutions. This possible restriction of solutions has been previously neglected in computational approaches. However, exploiting the existence of these sets can significantly reduce the time needed to find a minimum power dominating set and guarantee its optimality, thereby improving the

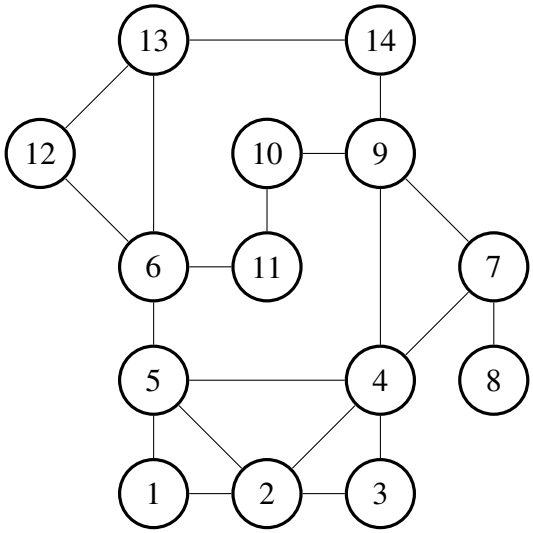

Figure 2.1 : A graphical representation of the IEEE 14 Bus system. Electrical networks are often relatively sparse and contain many low degree vertices.

runtime performance of the algorithms introduced in this thesis, as well as the performance of previously developed work such as Integer Program 1.4.

2.1 The Junction Vertex Partition

The vertices of a connected graph G can be partitioned into a sets of degree three or more vertices, which I refer to as *junctions*, and a set of path inducing subgraphs which may be each adjacent to one or more of the junctions. I will refer to these path inducing subgraphs as *junction paths*. The formal descriptions of these sets are given as follows:

Definition 2.1. Junction Vertex Partition:

$$J(G) := \{v \in V : \delta(v) \geq 3\},$$

$$\mathscr{P}(G) := \{P \subseteq V : P \text{ is a component of } G[V \backslash J(G)]\}.$$

From the definitions of these sets, it can be seen that each vertex in V is either contained in $J(G)$ or is contained in exactly one junction path $P \in \mathscr{P}(G)$. Thus, the sets

$J(G), P_1, \ldots, P_{|\mathscr{P}(G)|}$ form a partition of the vertices of G. Several observations regarding these sets can be made in the case that G is connected and $J(G)$ is non-empty:

Observation 2.1. For any connected graph G such that $J(G)$ is non-empty, each junction path P in $\mathscr{P}(G)$ is such that $N(P) \subseteq J(G)$. Additionally, either $|N(P)| = 1$ or $|N(P)| = 2$.

Observation 2.1 follows because no paths P_i, P_j can be adjacent to one another. This implies that the only possible neighbors for these sets are those contained in $J(G)$. Additionally, if $|N(P)| \geq 3$, then P must contain a junction. Since G is assumed to be connected and to contain a degree 3 or higher vertex, each path P is in the same component as a high degree vertex. If a path P does not have a neighbor, then it forms its own component in G, contradicting the connectedness of the graph. Thus $|N(P)| > 0$ and $|N(P)| < 3$.

Observation 2.2. For any connected graph G such that $J(G)$ is non-empty, and for any junction path P in $\mathscr{P}(G)$ such that $|N(P)| = 1$, either one or two vertices in P are adjacent to a vertex that is not in P.

This second observation provides a characterization for the junction paths based on their neighborhoods. Thus, a trichotomy for junction paths can be considered. The first type are *pendant paths* and have exactly one vertex adjacent to a vertex not in P. The second type are referred to as *pendant cycles*, and contain two vertices which are adjacent to some vertex u not contained in P. While the vertices in P do not induce a cycle in G, $P \cup \{u\}$ does induce cycle in this case. Finally, the third category of junction paths can be called *joining paths*. These junction paths are adjacent to two distinct junctions in $J(G)$. Two of these categories appear in the IEEE 14 examples shown earlier, as illustrated in Figure 2.2.

Vertices in $V(G)$ can be partitioned in linear time, and the paths in $\mathscr{P}$ can classified in linear time as well. Relevant to the discussion of zero forcing forts, any two pendant paths that neighbor the same junction vertex form a fort. Likewise, any two joining paths with the

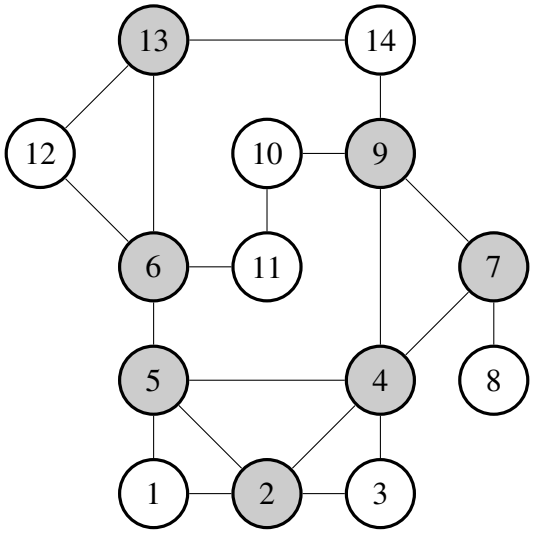

Figure 2.2 : The Junction Vertex Partition for the graph representation of the IEEE 14 Bus System. Shaded vertices are in $J(G)$ and maximal connected sets of unshaded vertices form junction paths. One pendant path and five joining paths can be seen in this example.

same neighbors also form a fort, and additionally each pendant cycle forms a fort. Efficient categorization of pendant paths allows for some forts and fort neighborhoods to be quickly detected. In the case of fort neighborhoods, the sets obtained this way correspond to distinct minimal fort neighborhoods. The algorithms introduced in 4 can make use of this procedure to find a partial list of constraints and improve their overall performance.

The following theorem makes use of the categories describing junction paths to construct a power dominating set comprised solely of a vertices in the set $J(G)$. Initially starting from any power dominating set in G, vertices not in $J(G)$ can be removed and replaced with vertices in $J(G)$ as needed. Furthermore, the cardinality of the constructed set can be verified to be no larger than the original power dominating set.

Theorem 2.1. For every power dominating set S in a connected graph G such that $J(G) \neq \varnothing$, there exists a set $S' \subseteq J(G)$ such that S' is a power dominating set in G and $|S'| \leq |S|$.

Proof. Let S be a power dominating set in G, such that $S \cap (V(G) \backslash J(G)) \neq \varnothing$, and let $S' = S$. Then let $\mathscr{P} = \{P \in \mathscr{P} : P \cap S \neq \varnothing\}$. For each $P \in \mathscr{P}_S$: remove all vertices in P from S', and if $S' \cap N(P) = \varnothing$ then add a vertex from $N(P)$ to S'. By applying the power domination color changing rule to S' after at most $t = \max_{P \in \mathscr{P}_S} |P|$ applications of the second step, all vertices in $N[S]$ are colored. Since this set of colored vertices contains a set that is assumed to be zero forcing, by Proposition 1.1 it must also be a zero forcing set. Thus $cl_P(S') = V(G)$, so S' is power dominating. By construction, $S' \subseteq J(G)$, and $|S'| \leq |S|$. □

Corollary 2.1. For any connected graph G such that $J(G)$ is non-empty, there exists a set of vertices $S \subseteq J(G)$ such that S is a minimum power dominating set in G.

Proof. Let G be a connected graph such that $J(G)$ is non-empty. Theorem 2.1 can then be applied to any minimum power dominating set in G, ensuring the existence of a minimum power dominating set S' contained in $J(G)$. □

Additionally, this result offers a bound on the power domination number of connected graphs with at least one vertex of degree three or more. While this bound is not strong in the case graphs with minimum degree three or more, in case of tree or tree like graphs such as those associated with power grids, this bound may be tighter than other bounds in power domination literature.

Note that Corollary 2.1 is similar to the claim posed by Haynes et al. [33]. However, the authors there do not provide a proof, nor do they include the necessary assumption that the graph is connected. In the case that G is a graph with $J(G) \neq \varnothing$ and has a component without a degree three or more vertex, no power dominating set exists made solely of vertices in $J(G)$. Thus, the additional assumption is indeed needed. While this difference is a relatively

minor distinction for a theoretical upper bound on $\gamma_P(G)$, this detail is the difference between correctness and infeasibility for Model 4.2 in Chapter 4.

Observation 2.3. There exists a graph G with no zero forcing set S contained in $J(G)$.

Unlike power domination sets, the corresponding statement for Corollary 2.1 is not valid when zero forcing sets are considered. This is demonstrated with a simple counterexample. Consider a claw graph: the graph obtained by appending 3 leaves to a vertex. The set of all junctions in this graph is just the original vertex, which is not a zero forcing set. In Chapter 4, Corollary 2.1 can be used to motivate a dimension reduction for computational techniques introduced there. This observation shows that this result cannot be simply extended to zero forcing as well.

2.2 Zero Forcing Forts and the Junction Vertex Partition

In this section, I give two basic lemmas regarding the structure of forts, fort neighborhoods, and the sets given by the junction vertex partition. In essence, the sets given by the junction vertex partition, namely $J(G), P_1, \ldots, P_k$ where $P_i \in \mathscr{P}(G)$ for $i = 1, \ldots, k$, can be thought of as pieces adding up to create fort neighborhoods. This idea is formalized in the necessary and sufficient conditions for fort neighborhoods: Theorem 3.1 in Chapter 3. To show this however, several lemmas must be established. To ease notation, another definition is introduced; any vertex v in a junction path P is called a *header* of P if it is adjacent to a vertex in $J(G)$. Then, consider the following lemmas:

Lemma 2.1. Let F be a fort in graph G and P be a junction path such that $P \cap N[F] \neq \varnothing$. Then for any header v adjacent to junction u, at least one of u, v is in F. Additionally, any leaf w in P is in F.

Proof. First, I show that if $w \in P$ and w is a leaf, then v is in $N[F]$. Assume that w is a leaf in P, and to verify the contrapositive of this statement, assume that $v \notin F$. Then by the definition of a fort, any node adjacent to v cannot be in F. This argument can be repeated for each vertex in P, and then any junction adjacent to P (if one exists).

Next, consider the case where v is a header of P and adjacent to a junction u. Also proving this statement using its contrapositive, if neither v nor u are in F, then any other neighbor of v cannot be in F. Repeating this argument iteratively for each other member of P, and the other junction in $N(P)$ if one exists, it is shown that $P \cap N[F] = \varnothing$. □

This lemma shows that if a junction path intersects the neighborhood of a fort, then the ends of that junction path must also intersect the neighborhood of the fort. The next lemma shows that this relation between junction paths and the neighborhoods of forts is much stronger. In fact, if a junction path intersects the closed neighborhood of a fort at all, it must be entirely contained in closed neighborhood of the fort.

Lemma 2.2. For any graph G with a fort F and junction path P, if $N[F] \cap P \neq \varnothing$ then $P \subseteq N[F]$.

Proof. It can first be shown that for any graph G with fort F and junction path P, if $N[F]$ intersects P then for any adjacent u, v in P, at least one of u, v are in F. Again, by the same arguments used to shown Lemma 2.1, assume that neither u nor v are in F. Then any other vertices adjacent to u or v are also not in F. This argument is repeated for each vertex in P, and any junctions in $N(P)$. Thus if $P \cap N[F] \neq \varnothing$, then for any 2 adjacent vertices in P, at least one is in F. Alternatively, if P is made up of a single vertex and intersects $N[F]$, then clearly P is entirely contained in $N[F]$. In all cases, $P \subseteq N[F]$. □

Chapter 3

Zero Forcing Fort Neighborhoods

This chapter is focused upon investigating the neighborhoods of zero forcing forts. Simply referred to as fort neighborhoods, the relationship between these structures and the power dominating set problem is akin to that of zero forcing forts and the zero forcing set problem. Forts indicate areas within graphs that cannot be colored by either the second step in the power domination color changing rule or the zero forcing color changing rule. Since the first step in the power domination color changing rule colors all neighbors of the initially colored vertex set, it is possible for vertices in forts to be colored despite starting with no colored members. This is not the case for the zero forcing color changing rule. Because of this difference, the neighborhoods of these forts are as crucial to the power dominating set problem as the forts themselves.

The contents of this chapter are arranged as follows. In the first section, the structure of fort neighborhoods is investigated. A set of necessary and sufficient conditions for determining if a vertex set S is the neighborhood of a fort are then given. These conditions build upon the vertex partition scheme defined in Chapter 2, and show that a fort with closed neighborhood S can always be constructed if two easily verified properties are satisfied. Using these necessary and sufficient conditions, a proof is given showing that the vertices of any minimal fort neighborhood induce a connected subgraph. These two characteristics are powerful tools that can be utilized when reasoning about fort neighborhoods in graphs. Employing these tools, I then give an example of a graph which can be verified to contain a subexponential number of minimal fort neighborhoods.

Section 3.2 explores both the computational complexity and two practical options for detecting minimum weight fort neighborhoods in graphs. In Chapter 4, the minimum weight fort neighborhood problem becomes the a subproblem that must be repeatedly addressed in the separation algorithms introduced there. Here, the details of this subproblem are the main focus and techniques for detecting these structures are introduced.

3.1 The Structure of Fort Neighborhoods

I first present necessary and sufficient conditions for a set S to be a the neighborhood of a fort in a graph G. In addition to offering an efficient way to determine if a set is the neighborhood of a fort, these conditions offer insights into the fundamental nature of fort neighborhoods.

Theorem 3.1. Necessary and sufficient conditions for fort neighborhoods:
There exists $J_S \subseteq J(G), \mathscr{P}_S \subseteq \mathscr{P}(G)$ such that for $S := \big(\bigcup_{P_i \in \mathscr{P}_S} P_i\big) \cup J_S, J_N := \{v \in J_S : |N(v) \cap (V \backslash S)| \geq 1\}$,
the following properties hold:

1. For any junction v adjacent to a junction path $P \in \mathscr{P}_S$, v is in J_S,
2. For any junction u in $J_N, |(N(u) \backslash J_N) \cap S| \geq 2$,

if and only if there exists a fort F in G, such that $N[F] = S$.

Proof. First, note that J can be partitioned into three disjoint sets: J_0, J_N, and $J \backslash J_S$, where $J_0 := \{j \in J_S : |N(j) \cap (V \backslash S)| = 0\}$.

To prove the sufficiency of these two conditions, suppose there exist $\mathscr{P}_S, J_S$ such that $S := \big(\bigcup_{P_i \in \mathscr{P}_S} P_i\big) \cup J_S$, and $J_S, \mathscr{P}_S$ satisfy $J_S \subseteq J(G), \mathscr{P}_S \subseteq \mathscr{P}(G)$, and Properties 1 and 2. Then, let $F := J_0 \cup \mathscr{P}_S$. It will suffice to show that $S = N[F]$, and that F is a fort.

$S = N[F]$ can be shown by set inclusions. If $v \in S$ either there exists a $P \in \mathscr{P}_S$ such that $v \in P, v \in J_0$, or $v \in J_N$. If $v \in P$ or $v \in J_0$, then $v \in F$. If $v \in J_N,$, then by Property 2, v is adjacent to at least 2 vertices in F. Next, to show that $N[F] \subseteq S$, note that if $v \in N[F]$ either $v \in F$, $v \in N(u)$, or $v \in N(P)$, for some $u \in J_0$, or $P \in \mathscr{P}_S$. All neighbors of u are in S, and by Property 1 any junction that is adjacent to P is in S. Thus $S = N[F]$.

F can be shown to be a fort by definition. That is, $\forall v \in V \backslash F,$ either $|N(v) \cap F| = 0$ or $|N(v) \cap F| \geq 2$. Let $v \in V \backslash F$. If $v \notin N(F)$, then $|N(v) \cap f| = 0$. If instead $v \in N(F)$, then there exists a $w \in F$ that is adjacent to v. By property 1 and the definition of J_0, any vertex that is adjacent to a path in $\mathscr{P}_S$ or a junction in J_0 must be in S. Thus $v \in S \backslash F$, so $v \in J_N$. Additionally, consider the following identity:

$$(N(v) \backslash J_N) \cap S = N(v) \cap (S \backslash J_N) = N(v) \cap F.$$

Making use of this identity and the assumption property 2 is satisfied:

$$|(N(v) \backslash J_N) \cap S| = |N(v) \cap F| = |(N(v) \backslash J_N) \cap S| \geq 2.$$

By definition, F is a fort.

To show the necessity of these conditions, assume that F is a fort in G. Then, let $J_S := J \cap N[F]$, and $\mathscr{P}_S := \{P \in \mathscr{P} : P \cap N[F] \neq \varnothing\}$. It will suffice to show that $J_S \subseteq J$, that $\mathscr{P}_S \subset \mathscr{P}$, and that Properties 1 and 2 hold. Clearly, $J_S \subseteq J$. By Lemma 2.2, no junction path can be partially contained in the neighborhood of a fort; $\mathscr{P}_S \subseteq \mathscr{P}$.

To prove property 1, note that $\forall P \in \mathscr{P}_S, P \subseteq N[F]$. Thus by Lemma 2.1, for any $v \in N(P_i)$, v must be in $N[F]$, so Property 1 is satisfied. Property 2 can be shown to hold by using the set equivalence identity shown earlier: $N(v) \cap F = (N(v) \backslash J_N) \cap S$. Since F is assumed to be a fort,

$$|(N(v) \backslash J_N) \cap S| = |N(v) \cap F| \geq 2.$$

Thus Property 2 is also satisfied, concluding the proof. □

Using the conditions presented by Theorem 3.1, the structural features of fort neighborhoods can be better understood. In particular, the following theorem describes an additional necessary condition for all minimal fort neighborhoods:

Theorem 3.2. For any graph G, if $M \subseteq V(G)$ is a minimal fort neighborhood, then $G[M]$ is connected.

Proof. This statement can be shown by its contrapositive. Let G be a graph, and M be a fort neighborhood in G such that $G[M]$ is disconnected. It can be shown that each component of $G[M]$ forms a fort neighborhood. Let M_0 be the vertex set of a component in $G[M]$. Then for any $P \in \mathscr{P}(G)$ such that $P \cap M_0 \neq \varnothing$, P is contained in M_0. Since $P \subset M$, by Theorem 3.1 $N(P) \subset M$, and thus $N(P) \subset M_0$. In G, M_0 satisfies the first of the two necessary and sufficient conditions for vertex sets to be fort neighborhoods. For any vertex $v \in J(G) \cap M_0$, if v is adjacent to a vertex not in M then since $v \in M$ and M is a fort neighborhood in G, v must be adjacent to at least two vertices that are in $J(G)$ with a neighborhood entirely contained in M or contained in some $P \in \mathscr{P}(G)$. Thus by Theorem 3.1, M_0 is a fort neighborhood in G and M is not minimal. □

In particular, these two theorems are useful for reasoning about minimal fort neighborhoods that may be present in general graphs. An example of such use can be seen in the proof of the next theorem. This theorem demonstrates that the number of distinct minimal fort neighborhoods in graphs may increase at a subexponential rate with respect to the order of graphs. To do so, I introduce the graph parameter $m_{min}(G)$, to denote the number of distinct minimal fort neighborhoods in a graph G. Then, to prove a lower bound on the potential number of minimal fort neighborhoods in sequence of graphs, consider the following theorem:

Theorem 3.3. There exists a family of graphs for which $m_{min}(G_k) = O(2^{\sqrt{v(G_k)}})$.

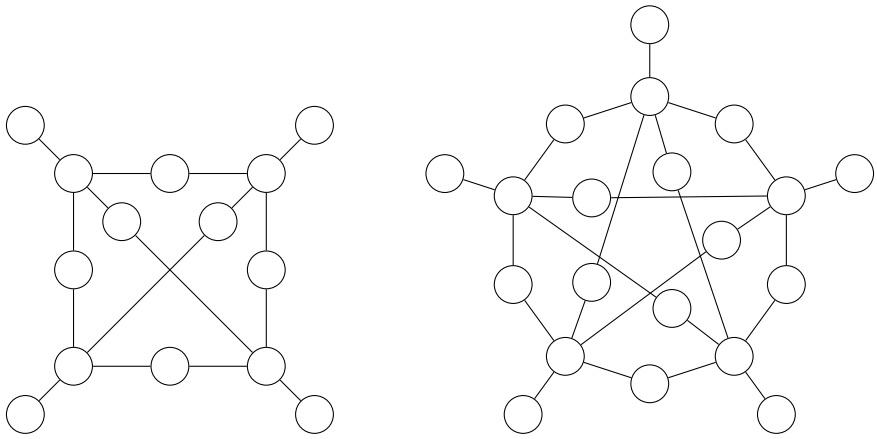

Figure 3.1 : Examples of construction used for Theorem 3.3, where $k = 4, 5$

Proof. This can be shown by direct construction. Consider the family of graphs generated by the following procedure. For any positive integer $k \geq 3$, define G_k to be the graph obtained by subdividing each edge of the complete graph K_k, and appending a leaf to each node originally in $V(K_n)$. Examples of the construction for $k = 4, 5$ are given as Figure 3.1.

Next, consider the vertex set M obtained by fixing a cycle in $V(G_n)$, removing a vertex from said cycle to form a path inducing vertex set, and the leaf adjacent to each end of the path. The set of degree 1 and 2 vertices in M form a fort for which M is the closed neighborhood.

M can be shown to be minimal by contradiction. Assume that M is not minimal. Then there exists some $M_0 \subset M$ such that M_0 is a minimal fort neighborhood. By Theorem 3.2, any minimal fort neighborhood induces a connected subgraph. Since $G[M]$ is a path, $G[M_0]$ is also a path graph. Then, if M_0 contains both degree one vertices in M, then since M_0 is connected, either $M_0 \subseteq M$ or M_0 contains at least one vertex not in M. The second option contradicts $M_0 \subset M$. Alternatively if M_0 does not contain at least one of the degree one vertices in M, then it is either an isolated node, or one of its end vertices is not a leaf in G_k. There are no isolated nodes in G_k with $k \geq 3$, so M_0 cannot be a single node.

If one of the end vertices of M_0 is not a leaf then that end vertex is either a node originally in K_k, thus has a neighbor that is adjacent to only one member of M_0 violating the second condition of Theorem 3.1, or the end vertex was added at the subdivision step and has only one adjacent member of M_0, violating the first condition of Theorem 3.1. Thus, M_0 is not a fort neighborhood, so M is indeed minimal.

Finally, the number of vertex sets in G_k that can be obtained by this process can be counted as follows:

$$\begin{aligned} m_{min}(f) \geq & \sum_{i=3}^{k} \binom{k}{i} (i) \\ = & \sum_{i=0}^{k} \binom{k}{i} (i) - O(n^3) \\ = & 2^k - O(n^3) \\ = & O(2^k) = O(2^{\sqrt{v(G_k)}}). \end{aligned}$$

□

3.2 The Detection of Fort Neighborhoods

Finding fort neighborhoods is the backbone of the separation algorithms presented in this thesis. Shown in Chapter 4, these sets correspond to constraints in the master problem. Quickly identifying these violated constraints can significantly improve runtime performance. In this section, I discuss the computational complexity of identifying these violated constraints and the related subproblem in the zero forcing separation algorithm presented by Fast [28]. Then, I provide both a polynomial time heuristic and an exponential time exact algorithm for generating violated constraint sets.

3.2.1 Computational Complexity

The decision version of the min weight fort neighborhood problem can be formally stated as follows:

Problem: Min Weight Fort Neighborhood
Instance: Graph G, Vertex Weights $W \in [0,1]^n$, Integer k
Question: Does G contain a fort neighborhood of weight less than k?

In the case that $k = 1$, this problem exactly defines the subproblems that the separation algorithm in Chapter 4 must solve. The remaining parameters are given by the graph G that is considered and fractional vertex weights in the interval $[0,1]$, which are determined by the optimal solution of a LP relaxation of the master problem. Presently, I have been unable to resolve the computational complexity of the above problem. However, the following analysis on a related problem may still provide some light on the complexity of this problem, as well as the tractability of possible algorithms that could be developed. Consider the problem:

Problem: Restricted Min Weight Fort Neighborhood
Instance: Graph G, Vertex v, Vertex Weights $W \in [0,1]^n$, Integer k
Question: Does G contain a fort neighborhood of weight less than k that contains v?

Intuitively, one approach for finding fort neighborhoods of low vertex weight is to construct a fort neighborhood from a specific vertex or a set of vertices, potentially improving a solution with some iterative process through a local search. In this case, a vertex or set of vertices may be known to be in the fort neighborhood that is being constructed. This type of approach corresponds to the restricted min weight fort neighborhood problem listed above.

However, this problem can be verified to be NP-Complete in general:

Theorem 3.4. The Restricted Min Weight Fort Neighborhood problem is NP-Complete.

Before proving this theorem, it is easier to see that this is also true for the analogously posed question regarding forts. The computational complexity of the decision problem for minimum weight forts is also of interest for the zero forcing separation algorithm introduced by Fast [28]. However, this problem is also unaddressed in previous literature. Here I provide a polynomial reduction for the restricted minimum weight fort problem to the well known 3-SAT problem. Consider the following problem statements and theorem:

Problem: Restricted Min Weight Fort

Instance: Graph G, Vertex v, Vertex Weights $W \in [0,1]^n$, Integer k

Question: Does G contain a fort of weight less than k that contains v?

Problem: 3-SAT

Instance: Boolean formula f in conjunctive normal form

Question: Is there a variable assignment such that f is satisfied?

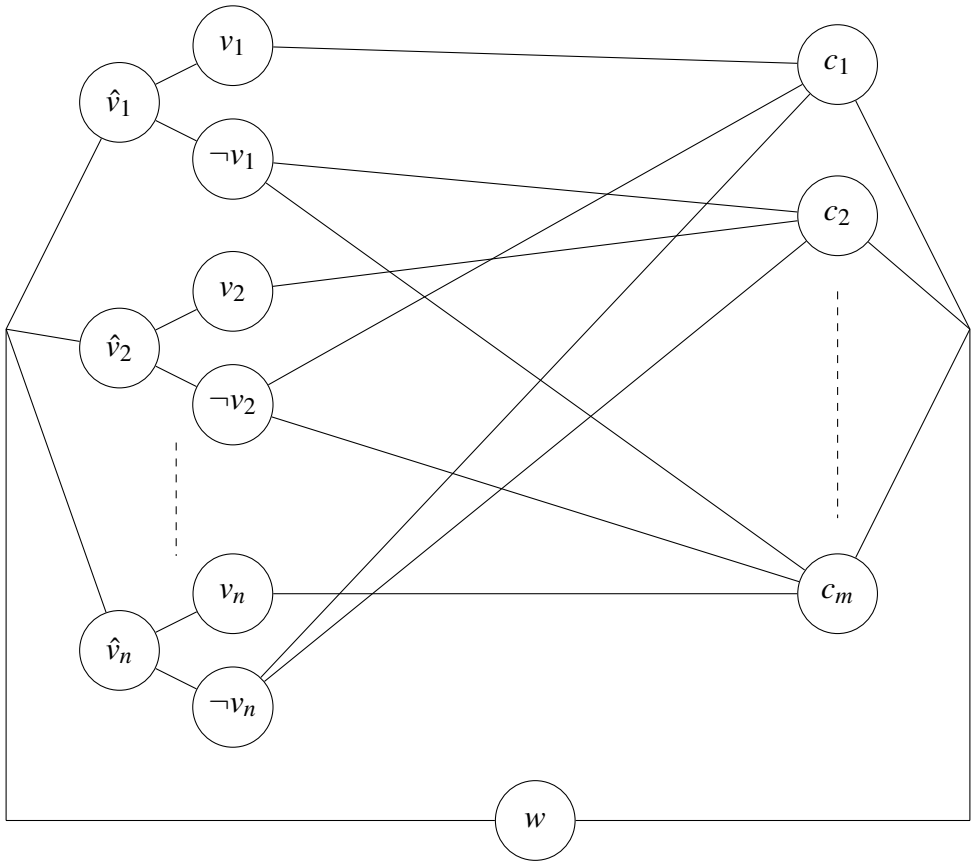

Figure 3.2 : Visual representation of the construction used in proof of Theorem 3.5

Theorem 3.5. The Restricted Min Weight Fort Problem is NP-Complete.

Proof. Any candidate set of vertices, F in a graph G, can be verified to be a fort of weight less than k by checking each vertex in $V(G)\backslash F$ to verify that it is not adjacent to exactly 1 member of F and by checking that the total weight of vertices in F is less that k. Thus, the Restricted Min Weight Fort problem is in NP.

To show that this problem is also NP-Hard, a construction given a general 3-SAT problem with boolean formula f with clauses $C_1, \ldots, C_m$ and variables $V_1, \ldots, V_n$ can be built. Let G be the graph obtained by the following: Add a vertex c_i for each clause C_i in f, vertices $v_i, \neg v_i, \hat{v}_i$ for each variable V_i in f, and a vertex w. Then, add any edge $v_i c_i$ if V_i appears in

clause C_i. Likewise, add the edge $\neg v_i c_i$ if $\neg V_i$ appears in clause C_i. Next, add edges $v_i\hat{v}_i, \neg v_i\hat{v}_i$ for each i in $1,\ldots,n$. Finally, add edges $c_j w$, $\hat{v}_i w$ for $i = 1,\ldots,n$, $j = 1,\ldots,m$. Since the total number of vertices and edges built in this construction is linear with respect to the size of f, this construction can be built in a polynomial number of steps. A visualization of this construction is provided in Figure 3.2.

In this newly constructed graph G, assign vertex weights as follows: let $v_i, \neg v_i$ have weight 1 and $c_j, \hat{v}_i$ have weight $n+1$ for $i = 1,\ldots,n$ and $j = 1,\ldots,m$. Let w have weight 0. Then, consider the Restricted Minimum Weight Fort Problem with mandatory vertex w, the vertex weights assigned for G, and the integer $n+1$. Next I will show that f is satisfiable if and only if G has a fort F containing w of total vertex weight less than $n+1$. Moreover, the $v_i, \neg v_i$ vertices contained in F indicate a variable assignment that satisfies f.

First, assume that f is satisfiable with variable assignment V_f. Then it is easy to see that the union of w and the corresponding $v_i, \neg v_i$ in V_f form a fort, call it F, in G. Since there are a total of n variable assignments and m has weight 0, this set has weight n.

Alternatively, assume that F is a fort in G that contains w and has total vertex weight less than $n+1$. Then, F cannot contain any of the $c_j, \hat{v}_i$ vertices as they are of weight $n+1$. Since they are adjacent to w, which is a member of fort F, they must each be adjacent to at least one other member of F. Thus for each $v_i, \neg v_i$ pair, at least one vertex is in F. By the pigeonhole principle, if any $v_i, \neg v_i$ pair are both contained in F, then another pair is entirely excluded from F. Thus, each $v_i, \neg v_i$ pair must have exactly one member contained in F. The $v_i, \neg v_i$ included in F therefore form a valid variable assignment for f.

Finally, this variable assignment is shown to satisfy f. Since each c_i is adjacent to the fort F but not in F, it must also be adjacent to another member of F. Each c_i node is only adjacent to w and vertices corresponding to variable assignments in its corresponding C_i clause. Thus, the other member of F that c_i is adjacent to must also correspond to a variable

assignment in C_i. The $v_i, \neg v_i$ included in F encode a variable assignment that satisfies f. Thus, the Restricted Min Weight Fort Problem is NP-Complete. □

A similar construction can also be used to prove Theorem 3.4:

Proof of Theorem 3.4. The conditions given by Theorem 3.1 can be used to verify in a polynomial number of steps, that a set of vertices is in fact the neighborhood of a fort. Checking that a candidate set has total vertex weight less than or equal to k and contains a specified vertex can also be done in a linear number of steps. Thus, the Restricted Min Weight Fort Neighborhood Problem is in NP.

Next, a construction similar to the Restricted Min Weight Fort construction can be considered. For boolean formula f, with variables V_i for $i = 1, \ldots, n$ and clauses C_j for $j = 1, \ldots m$, let G be the graph obtained by the following steps: let there be a vertex c_j for each clause C_j, vertices $v_i, \neg v_i, \hat{v}_i$ for each variable V_i, and additional vertices w_{man}, w_{fort}, w_{out}, and w_0. Then add edges $v_i\hat{v}_i$, $\neg v_i\hat{v}_i$, $\hat{v}_i w_{fort}$, and $\hat{v}_i w_{out}$ for each variable V_i and edges $c_j w_{fort}$ and $c_j w_{out}$ for each clause C_j. Next, add an edge between vertices v_i and c_j for each variable assignment V_i appears in a clause C_j, for each clause in f. Likewise, add edge $\neg v_i c_j$ for each variable assignment $\neg V_i$ that appears in clause C_j. Finally, add edges $w_{out}w_{man}$, $w_{fort}w_{man}$, and $w_0 w_{man}$. In totus, this construction contains $m + 3n + 4$ vertices and $5m + 4n + 3$ edges; it can be constructed in a polynomial number of steps with respect to the size of the 3-SAT problem that is encoded. A visualization for this construction is provided in Figure 3.3.

Vertex weights for G can be assigned as follows. Let vertex w_{out} have weight $n + 1$ and vertices $v_i, \neg v_i$ have weight 1 for each variable V_i in f. Let all other vertices have weight 0. Next I will show that f is a satisfiable formula if and only if G has a fort neighborhood containing w_{man} of total vertex weight less than $n + 1$. Moreover, the variable assignment

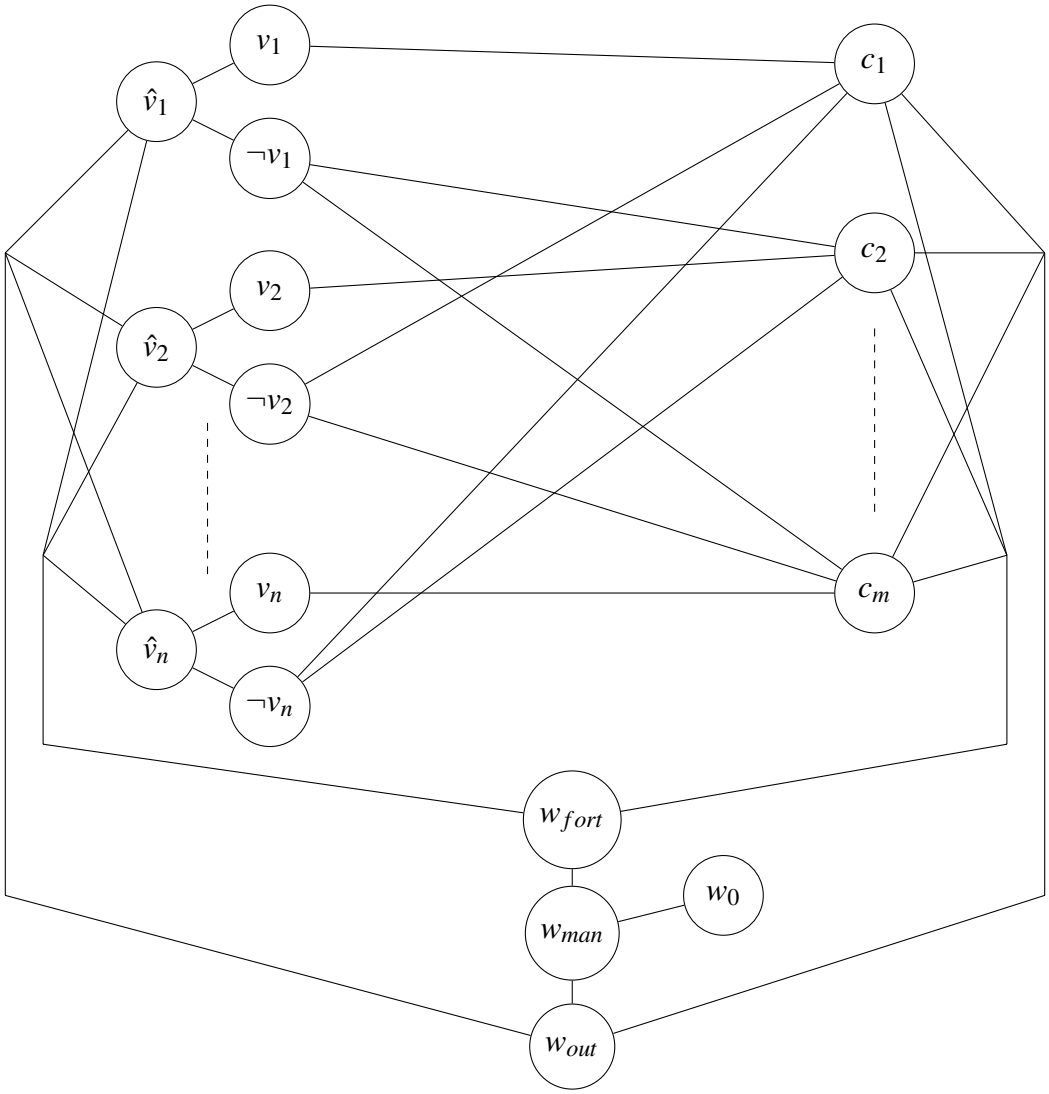

Figure 3.3 : Visual representation of the construction used in proof of Theorem 3.4

vertices that this fort neighborhood contains encode a variable assignment that satisfies f.

First, assume that f is satisfied by some variable assignment V_f. The union of $\{w_{fort}, w_0\}$ and the set of vertices $v_i, \neg v_i$ corresponding to variable assignments in V_f form a fort in G. Then the union of $\{w_{man}, w_{fort}, w_0\}$, the corresponding variable assignment vertices $v_i, \neg v_i$ in V_f, $\hat{v}_i$ for each variable V_i, and c_j for each clause C_j forms the neighborhood of that fort. This fort neighborhood includes w_{man} and has total vertex weight n.

Alternatively, assume that M is the neighborhood of a fort in G, contains vertex w_{man}, and has total vertex weight less than $n+1$. Since w_{man} is in M but is also adjacent to a vertex not in M, by the second condition of Theorem 3.1 the remaining two neighbors of w_{man} must have neighborhoods entirely contained in M. Equivalently, since w_{man} is in the neighborhood of a fort but as adjacent to a non member of M, its remaining neighbors w_{fort}, w_0 must be in the fort itself. Each clause vertex c_j must be adjacent to another member of this fort as well, since these vertices are also adjacent to w_{fort} and w_{out}, as must each $\hat{v}_i$ vertex. By the same pigeonhole principle argument presented in Theorem 3.5, each $\hat{v}_i$ is adjacent to no more than one vertex in $\{v_i, \neg v_i\}$ contained in M. Thus, each variable V_i has exactly one corresponding variable assignment vertex in M. This forms a valid variable assignment for M. Then since each clause vertex c_j must be adjacent to at least one additional member of M and is only adjacent to variable assignments that satisfy it, this variable assignment must satisfy f. Thus, fort neighborhoods with the assumed conditions encode satisfying variable assignments to the original 3-SAT problem. The Restricted Min Weight Fort Neighborhood Problem is NP-Complete. □

The construction used to prove Theorem 3.4 can also be used to show that finding a minimum weight fort neighborhood M with a vertex contained in M and vertex contained not in M is NP-Complete to solve in general. The hardness of this problem in the case that no vertex is required to be contained in the fort neighborhood set is unknown.

3.2.2 Integer Program Approach

Given the necessary and sufficient conditions for a vertex set to be the neighborhood of a fort presented as by Theorem 3.1, an integer program can be used to identify minimum weight fort neighborhoods. This presents a first option for finding minimum weight fort neighborhoods. While this approach is based upon an exponential time algorithm, computational results given in Chapter 5 indicate that this approach is computationally effective for the typical problems seen in power domination literature. This model is presented as Integer Program 3.1.

When a fort neighborhood is found using Integer Program 3.1, the model's solution also encodes a fort with that neighborhood. In addition, each vertex in graph G is considered to be a member of one of three sets. The set O is made up of vertices that are not contained in the neighborhood of the discovered fort. The set B are the vertices in $V(G)$ that are in the neighborhood of the fort, but are not in the fort itself. Finally, the set I indicates which vertices are members of the candidate fort. Using the necessary and sufficient conditions for fort neighborhoods, members of these sets must each satisfy conditions that can be verified with a system of constraints.

Finally, this model incorporates vertex weights in its cost function as c_v for $v \in V(G)$ and assumes that the graph G is connected and contains a vertex of degree 3 or more. If G is not connected or does not contain a vertex of degree 3 or more, it can be efficiently detected. In this first scenario, the low degree component is itself a fort neighborhood and the only minimal fort neighborhood with vertices in that component. Components with at least one degree 3 or more vertex satisfy the assumptions of this model and can be solved directly. Without loss of generality then, we can assume that G is connected and has a vertex of degree at least 3.

Integer Program 3.1. Min Fort Neighborhood Model

$$\begin{aligned}
\text{Minimize} \quad & \sum_{v \in V} c_v(I(v) + B(v)) \\
\text{Subject To} \quad & \sum_{v \in V} I(v) + B(v) \geq 1 && (1) \\
& I(v) + B(v) + O(v) = 1 \qquad \forall v \in V && (2) \\
& B(v) = 0 \qquad \forall v \in V : \deg(v) \leq 2 && (3) \\
& |N(v)| * I(v) + \sum_{u \in N(v)} O(u) \leq |N(v)| \qquad \forall v \in V && (4) \\
& \sum_{i \in N(v)} I(v) \geq 2B(v) \qquad \forall v \in V : \deg(v) \geq 3 && (5) \\
& I, B, O \in \mathbb{B}^{|V|}
\end{aligned}$$

In the Min Fort Neighborhood Model, constraints intuitively enforce the following features. Constraint (1) ensures that the empty set is not a valid solution. Constraint (2) forces each vertex to be a member of exactly one of the I, B, O sets. Constraint (3) ensures that no vertices that are not junctions in G are contained in B. The fort constructed in the proof of Theorem 3.1 shows that if a set is the neighborhood of a fort, then a fort F with only junctions in $N[F]$ exists. Next, Constraint (4) ensures that vertices in the fort are not adjacent to any vertices not adjacent to the neighborhood of the fort. Finally, Constraint (5) requires that each vertex in the open neighborhood of the fort is adjacent to at least 2 members of the fort. This is formally shown in the following theorem.

Theorem 3.6. The optimal solution of Integer Program 3.1 is a minimum weight fort neighborhood for graph G.

Proof. To prove this theorem it suffices to show that solutions for the model each correspond to fort neighborhoods in G, that each fort neighborhood corresponds to a feasible solution for the model, and that the objective function correctly optimizes the minimum weight fort neighborhood.

Let O, B, I be the solution for Integer Program 3.1. Since these cases can be easily detected and resolved as discussed before, without loss of generally assume that G is connected and contains at least one vertex of degree 3 or more. Since O, B, I are binary vectors and Constraint (2) requires that for each v in $V(G)$ exactly one of $O(v), B(v), I(v)$ is non-zero, these sets form a partition for the vertices of G. By Constraint (1), $I \cup B$ cannot be an empty set. Next, it is shown that $I \cup B$ forms the vertex set of a fort neighborhood by verifying that this union satisfies the necessary and sufficient conditions given in Theorem 3.1.

To see that the first property of Theorem 3.1 is satisfied, consider any junction path P in G. Constraint (3) requires that each vertex in P is either in I or O. Constraint (4) ensures that any vertex in both P and I cannot have neighbors are in O. Thus, if any vertex in P is in I, then all vertices in P are in I, and all junction vertices adjacent to P are in I or B. Therefore $I \cup B$ satisfies Property 1.

To see that the second property of Theorem 3.1 is satisfied, consider any junction $v \in I \cup B$. If v is in I then all neighbors of v are in I or B and the assumptions of the second property do not apply to v. In that case that v is in B, Constraint (5) requires that v is adjacent to at least two vertices in I, thus Property 2 is also satisfied so $I \cup B$ forms a fort neighborhood.

Alternatively if M is a fort neighborhood in G, $V(G)$ can be partitioned in the following way: for each vertex in $V(G) \backslash M$, let $v \in O$. For each degree 2 or less vertex in M let $v \in I$. For each degree 3 or more vertex v in M such that $N(v) \subset M$, let $v \in I$, and let v be in B otherwise. These assignments satisfy all constraints in Integer Program 3.1 so each fort neighborhood corresponds to a feasible solution for the model. Since the objective function of this model is just the total vertex weight of the identified fort neighborhood, solutions corresponding to minimum weight for neighborhoods are optimal solutions. □

3.2.3 Random Heuristic Approach

The integer program approach for finding violated constraints is guaranteed to find fort neighborhoods with total vertex weight less than one if they exist, but it may be more computationally efficient to find violated constraints by using a heuristic with worst-case polynomial complexity. To this end, the following heuristic is presented.

The heuristic presented here works by randomly generating vertex sets S such that $V(G)\backslash S$ has total weight less than k. Then if S is not a power dominating set, the complement of the power dominating closure of S forms a fort in G. Moreover, its closed neighborhood does not intersect the closed neighborhood of the initial set S, and thus has total weight less than k as desired.

Algorithm 3.1 Fort Neighborhood Random Heuristic

$W(v) =$ Non negative valued weights
$R = \{v \in V(G) : W(v) > 0\}$
$R_{sum} =$ sum of $W(v)$ for each v in R
if $R_{sum} < k$ **then**
    return $V(G)$
$i = 0$
while $i < MaxIterations$ **do**
    $sum = 0;\ R' = R;\ S = \varnothing$
    while $sum \leq R_{sum} - k$ **do**
        Randomly select r in R'
        $S = S \cup r$
        $sum = sum + W(r)$
        $R' = R' \backslash \{r\}$
    if $V \backslash cl_P(S) \neq \varnothing$ **then**
        return $N[V \backslash cl_P(S)]$
    $i = i + 1$
return No solution found

In the following theorem, I formally show that the fort neighborhoods identified by this algorithm have total vertex weight less than k. In addition, I show that this algorithm has

worst case complexity $O(n^3)$.

Theorem 3.7. Program 3.1 returns a fort neighborhood of weight less than k or nothing with worst case $O(n^3)$ time complexity.

Proof. Let G be a graph, $MaxIterations \geq 1$, $W : V \to [0, \infty)$. Then due to line 4 if $R_{sum} < k$, then at line 5 the algorithm returns $V(G)$, which is trivially a fort neighborhood with weight less than k. Assume instead that $R_{sum} \geq k$. At each iteration of the while loop in line 7, S is initialized to be $\varnothing$, R' to be R, and $sum = 0$. Then, with each application of the while loop at line 9 an element is removed from R' and added to S, and sum is updated. This continues until $sum > R_{sum} - k$. At most n elements can be exchanged by this process; since $R_{sum} < k$ this while loop executes at least once and terminates when or prior to when all elements have been exchanged.

When the while loop at line 9 is exited, $sum > R_{sum} - k$. The IF statement at line 14 evalutes as true if and only if $V \backslash cl_P(S) \neq \varnothing$. Thus if a vertex set is returned at line 15, $V \backslash cl_P(S) \neq \varnothing$. By definition S is not a power dominating set and $V \backslash cl_P(S)$ is a fort. Since $N[V \backslash cl_P(S)] \subseteq V(G) \backslash S$, the total weight of $N[V \backslash cl_P(S)] < k$. Thus, if a vertex set is returned at line 15, it must be a fort neighborhood of weight less than k.

To verify the time complexity of Algorithm 3.1, note that lines 1 through 6 each execute in linear or constant time. Additionally, the while loop at line 7 executes a constant number of times as specified by the user parameter $MaxIterations$. The while loop at line 9 then executes a $O(n)$ times. Since $cl_P(S)$ can be calculated in $O(n^2)$ time, the overall time complexity of this heuristic is worst-case $O(n^3)$. □

In the next chapter, I show a way that this program can be utilized for generating violated constraints for the Linear Program (LP) relaxation of Integer Program 4.1. Specifically, this is done by setting $k = 1$ and the vertex weight function W to be the vertex weights of

the fractional optimal solution for the separation algorithm's master problem. While this program is not guaranteed to find an uncovered fort neighborhood, even in one exists, it has worst-case polynomial computational complexity. If no set is found after a maximum number of iterations is reached, Integer Program 3.1 can be used to verify that none exist or identify an undetected violated constraint. For large problem instances, this heuristic may outperform the approach offered by solely using Integer Program 3.1.

Chapter 4

Separation Algorithms for Power Domination

The structure of power dominating sets is intimately related to the presence neighborhoods of zero forcing forts. The work in this chapter formally explores this idea, and relates the results of Chapter 3 for use in practical computations. For example, a large portion of Chapter 3 was devoted to deriving necessary and sufficient conditions for sets to be fort neighborhoods. In this chapter these conditions are exploited to create methods for finding fort neighborhoods via integer programs or random heuristic methods.

The contents of this chapter are presented as follows. First, a set cover formulation for power domination is given and proven to equivalently describe solutions of the power dominating set problem. For practical implementation, a separation algorithm for the power domination set cover formulation is then introduced. Both a general algorithm and an algorithm finding solutions of the type guaranteed by Corollary 2.1 are produced. The differences between these algorithms are detailed in Section 4.2.

4.1 Set Covering and Power Domination

As discussed in Section 1.4, Fast [28] proposes a set cover approach for the zero forcing set problem. The constraint used in this formulation correspond to zero forcing forts, each of which must be covered at least once by any set that is zero forcing. Here, I present an analogous approach for the power dominating set problem. In this formulation however, the closed neighborhoods of forts must be covered rather than the forts themselves. Let $\mathscr{M}$

denote the set of closed neighborhoods of forts in G. Consider the following model:

Integer Program 4.1. Power Domination Set Cover Model

$$\begin{aligned} \text{Minimize} \quad & \sum_{v \in V} s_v \\ \text{Subject To} \quad & \sum_{v \in M} s_v \geq 1 \qquad \forall M \in \mathscr{M} \\ & s_v \in \{0,1\}, \qquad \forall v \in V \end{aligned}$$

Due to the first step in the power domination color changing rule, having an initially colored vertex in the neighborhood of each fort in a graph is sufficient for a set to be power dominating. This follows because at the end of the first step in the power domination color changing rule at least one member of each fort is colored. Thus the resulting set is zero forcing, and the remaining uncolored vertices can be fully colored by repeatedly applying the second step of the power domination color changing rule. In addition, this condition is also necessary as shown by the following theorem:

Theorem 4.1. For any graph G, $S \subseteq V(G)$ contains at least one member of the closed neighborhood of every fort in G if and only if S is a power dominating set.

Proof. Let G be a graph and $S \subseteq V(G)$ contain at least one member of the closed neighborhood of every fort in G. For a contradiction, assume that S is not a power dominating set in G. Then the vertex set $F := V(G) \backslash cl_P(S) \neq \varnothing$. If any $v \in F$ is adjacent to exactly one member of $cl_P(S)$, then v can be colored by the second step of the power domination color changing rule, and v would be in $cl_P(S)$. Thus, each v is adjacent to either zero or two or more members of $cl_P(S)$. F is by definition a fort in G. Moreover, if any $u \in N[F]$ is in S, then a member of F would be colored by the first step of the power domination color changing rule. Thus no member of $N[F]$ is contained in S, a contradiction.

The sufficiency of this property can be shown by contrapositive. Assume that there is a fort F in G such that $S \cap N[F] = \varnothing$. Then $N[S] \cap F = \varnothing$. By the contrapositive statement

of Theorem 1.1 since there is a fort that is disjoint from $N[S]$, $N[S]$ is not zero forcing in G. Thus S is not power dominating in G. □

Corollary 4.1. Feasible solutions for the power domination set cover model, for a graph G, are characteristic vectors of power dominating sets in G, and the optimal value of the model is $\gamma_P(G)$.

As a corollary, I posit that optimal solutions for the power domination set cover model correspond to power dominating sets. Since any feasible solution for the model must cover all fort neighborhoods in G, by Theorem 4.1 the corresponding vertex set is power dominating. The optimal value of this model is then equal to the cardinality of the minimum power dominating sets in G.

The next model is a modification of Integer Program 4.1 which exploits the results given by Corollary 2.1. Corollary 2.1 guarantees the existence of a minimum power dominating set S such that $S \subseteq J(G)$ for any connected G with $J(G) \neq \varnothing$. As a result, if its assumptions hold then this model will attain the same optimum while potentially considering a smaller number of variables. I prove this claim in Theorem 4.2.

Integer Program 4.2. Junction Set Cover Model

$$\begin{aligned} \text{Minimize} \quad & \sum_{v \in J(G)} s_v \\ \text{Subject To} \quad & \sum_{v \in J(G) \cap M} s_v \geq 1 \qquad \forall M \in \mathscr{M} \\ & s_v \in \{0,1\}, \qquad \forall v \in J(G) \end{aligned}$$

Theorem 4.2. For a connected graph G with $J(G) \neq \varnothing$, the optimal value of the junction set cover model is $\gamma_P(G)$.

Proof. This theorem follows from Corollary 2.1 and Corollary 4.1. For any graph G, the optimal solution for power domination set cover model is the characteristic vector of a

minimum power dominating set. Any feasible solution for the junction set cover model indicates a power dominating set contained in $J(G)$, which is also a feasible for the power domination set cover model. Thus, the optimum of the junction set cover model is bounded below by the optimum of the power domination set cover model. By Corollary 2.1, if G is connected and $J(G) \neq \varnothing$ then there exists a power dominating set of cardinality $\gamma_P(G)$. The characteristic vector for this set is a feasible solution for the junction set cover model, so its optimum is $\gamma_P(G)$. □

For completeness, this theorem can be slightly strengthened. It applies to all graphs where each component of the graph contains a vertex of degree three or more. Here, I have chosen to state this theorem in a slightly weaker form for clarity.

While both of these models have optimum $\gamma_P(G)$, their assumptions regarding G differ. Additionally, the objective function of power domination set cover model can be easily changed to incorporate vertex weights. This model could then be use to consider a weighted version of the power dominating set problem. This is likely to be a practical concern as in the original application, the costs of installing sensors in power grids can change significantly from substation to substation. As a result, the costs of incorporating a sensor in some substations may be cheaper than in others, and merely finding minimum power dominating sets may not always provide the cheapest sensor array when implementation costs are considered. Nevertheless, in practice redundant, measurements and increased direct observations are also matters of concern. Placing sensors in highly connected substations, those corresponding to vertices in $J(G)$, is also desirable. Computational experiments indicate that the performance of solving the junction set cover model is significantly better than that of the power domination set cover model. This is intuitively clear as the junction set cover model can be thought of as a dimensional reduction of the regular set cover model. In this model, a largely reduced feasible region can be considered while the same optimum is

guaranteed to be reached.

Each of these models provably determines the power domination number of graphs satisfying their respective assumptions, however, identifying the sets corresponding to their constraints has not yet been addressed. To accomplish this, vertex sets of fort neighborhoods must be identified. In the next section I discuss several approaches for identifying these vertex sets. As demonstrated by Theorem 3.3, the number of constraints that must be satisfied may increase at a larger than polynomial rate in relation to the size of the graphs that are being considered.

Furthermore, if a constraint corresponding to a fort neighborhood is satisfied, any fort neighborhoods containing that neighborhood are also covered. Thus, it suffices to only consider minimal fort neighborhood constraints. This idea is supported by the findings of Balas and Ng provided in Chapter 1 as Theorem 1.2; constraints corresponding to non-minimal fort neighborhoods are never facet inducing inequalities. However, also shown by Theorem 3.3, the number of distinct minimal fort neighborhoods may increase with problem size at a faster than polynomial rate.

For large graphs, it is likely to be infeasible to explicitly find all constraints needed to fully define the feasible regions of either of the power domination set cover integer programs. In response to this issue, I have adopted to use a constraint generation approach. As in other separation algorithms, a master problem and a series of subproblems can be considered. In the algorithms presented in this chapter, the master problem is either of the previously defined set cover integer programs and the subproblems are defined as detecting the fort neighborhood cover constraints that solutions for the master problem violate.

In a general branch and cut framework for solving the power domination set cover integer program, the integrality constraints on the variables are relaxed and optimal fractional solutions satisfying a portion of the constraints are identified. This assigns each vertex a

weight in the interval $[0,1]$. Violated constraints then correspond to fort neighborhoods with total vertex weight strictly less than 1. When no more violated constraints can be found, fractional variables are branched upon until an integral optimal solution is found.

4.2 Separation Algorithms for Power Domination

4.2.1 General Separation Algorithm

Algorithm 4.1 General Power Domination Separation Algorithm

1: $\mathscr{M} \leftarrow \mathscr{M}_0$
2: Add set cover constraints for sets in $\mathscr{M}_0$ to Model 4.1
3: $S \leftarrow$ optimal solution for LP relaxation of Model 4.1
4: Set vertex weights for IP 3.1 to S
5: **while** IP 3.1 is feasible **do**
6: $\quad M \leftarrow$ optimal solution for IP 3.1
7: $\quad$ Add set cover constraint for set M to Model 4.1
8: $\quad S \leftarrow$ optimal solution for LP relaxation of Model 4.1
9: $\quad$ Set vertex weights for IP 3.1 to S
10: **if** S is fractional **then**
11: $\quad$ Branch on most fractional S_v, **go to** 5
12: **return** S

Algorithm 4.1 is a basic separation algorithm that finds a minimum power dominating set for a graph by repeatedly finding violated fort neighborhood constraints for fractional solutions to Integer Program 4.1. First, the constraint set $\mathscr{M}_0$ is added to Integer Program 4.1. In the case that no fort neighborhood constraints are known, $\mathscr{M}_0$ can be set to $\varnothing$.

Algorithm 4.1 then proceeds to find violated constraints by obtaining integral solutions for Integer Program 3.1 repeatedly until there are all fort neighborhood constraints are satisfied by S. At this point, S is a possibly fractional solution satisfying all fort neighborhood constraints. Then, if S is a non integral solution, the most fractional value of S is branched

upon and the algorithm returns to the violated constraint search. Eventually, an optimal integral solution is found and returned.

This algorithm is intended to be modular and can be adapted in several ways. First, the set $\mathscr{M}_0$ is in essence a warm start for the algorithm and can be initialized by a plethora of options. Next, the set cover model used for 4.1 was Integer Program 4.1, however Integer Program 4.2 can have also been used under certain assumptions. Finally, Integer Program 3.1 was used to find violated fort neighborhood constraints. The polynomial complexity random heuristic introduced in Chapter 3 can also be used to search for these constraints.

In the following subsections, these customization options are each investigated and discussed. Additionally, I provide more specific implementation details for the methods compared in Chapter 5, as well as the reasoning behind those decisions.

4.2.2 Initial Constraint Sets

The initial constraint sets for Algorithm 4.1 allow the separation algorithm to be adjusted to utilize information about easily found constraints. The approach used in my implementations stems from the Junction Vertex Partition discussed in Chapter 2. In polynomial time, the partition can be constructed, and junctions and junction paths can be combined to form fort neighborhood vertex sets. Examples of these combinations include any junction and two or more pendant paths adjacent to that junction, any junction and an adjacent pendant cycle, or any two junctions and two or more joining paths between them. It can be seen that in the case of the k-star graph that unless accounted for, this process could generate an exponential number of constraints. To avoid this, I used the following algorithm to initialize $\mathscr{M}_0$:

Algorithm 4.2 finds the number of pendant paths and pendant cycles adjacent to each junction $v \in J(G)$, as well as the number of joining paths between each pair of junctions

Algorithm 4.2 Initial Constraint Set Algorithm

1: $\mathscr{M}_0 \leftarrow \varnothing$
2: $pendpaths(v) \leftarrow \varnothing$ for $v \in J(G)$
3: $pendcycls(v) \leftarrow \varnothing$ for $v \in J(G)$
4: $joinpaths(v,u) \leftarrow \varnothing$ for (v,u) pair in $J(G)$
5: **for** $P \in \mathscr{P}(G)$ **do**
6: **if** P has no headers **then**
7: Append P to $\mathscr{M}_0$
8: **else if** P is a pendant path **then**
9: Append P to $pendpaths[v]$
10: **else if** P is a pendant cycle **then**
11: Append P to $pendcycls[v]$
12: **else if** P is a joining path between vertices v,u **then**
13: Append P to $joinpaths(v,u)$
14: **for** $v \in J(G)$ **do**
15: **if** $|pendcycls(v)| > 0$ **then**
16: Append $pendcycls(v)[0] \cup \{v\}$ to $\mathscr{M}_0$
17: **if** $|pendpaths(v)| > 1$ **then**
18: Append $pendpaths(v)[0] \cup pendpaths(v)[1] \cup \{v\}$ to $\mathscr{M}_0$
19: **for** (v,u) for (v,u) pair in $J(G)$ **do**
20: **if** $|joinpaths(v,u)| > 1$ **then**
21: Append $joinpaths(v,u)[0] \cup joinpaths(v,u)[1] \cup \{v,u\}$ to $\mathscr{M}_0$
22: **return** $\mathscr{M}_0$

$(v,u) \subseteq J(P)$. The algorithm avoids explicitly finding an exponential number of fort neighborhoods by only including one fort neighborhood of each type, for each vertex or pair of vertices in $J(G)$. Note that in case Model 4.2 is used these constraints result in either a required vertex or a set cover constraint over a pair of vertices, further reducing the dimension of the master problem.

It has also been shown that the power dominating set problem is solvable by polynomial time or even linear time complexity algorithms in some special cases including: trees [33, 15], block graphs [45], grid graphs [23], interval graphs [36], and circular-arc graphs [37]. In these cases it is likely that algorithms similar to Algorithm 4.2 could be devised to find a polynomial number of constraints that fully describe solutions for the power dominating set problem. If solutions of special forms such as optimal solution of degree 3 or more vertices are sought after, these constraints may reduce the set cover problem to a set of mandatory vertices, allowing certain graph families to be solved in polynomial time. This type of behavior is exhibited for some constraints found by Algorithm 4.2 if Integer Program 4.2 is used as the master problem.

4.2.3 Set Cover Models

While Algorithm 4.1 makes no assumptions upon the graph to which is is applied, graphs representing electrical networks are often connected and have at least one vertex of degree 3 or more. In the case that the network is not connected, it often the case that the components contain a vertex of degree 3 or more. These properties can be seen in the commonly considered IEEE test cases in power domination literature.

These properties can be exploited to significantly improve the runtime performance of Algorithm 4.1 by replacing Integer Program 4.1 with Integer Program 4.2. Doing so allows any set cover constraint C to be instead reduced to $C \cap J$. Additionally, variables are only

needed for vertices in J. Many of the commonly considered test cases feature networks with chains of degree 2 vertices which form long junction paths. By excluding these vertices as candidates for a minimum power dominating set, a significantly smaller set of feasible solutions for the master problem can be considered.

However, in the case that a minimum weight power dominating set is desired this modification may not be a good choice. If the cardinality of the power dominating set is not important but rather the weight of the vertices in that set are being minimized, this approach introduces cuts that may exclude optimal solutions. In this case the general set cover model should be used so that every power dominating set in the considered network has a corresponding feasible solution. In the Chapter 5, I present the computational results of when this modification is incorporated to the separation algorithm as well as when the general set cover model is considered.

Finally, the set cover model can be easily adapted to incorporate additional constraints that a solution must satisfy. As an example, in the case of the connected variant of power domination, solution vertex sets must induce connected graphs. Several sets of constraints have been introduced that force solutions to satisfy this additional requirement. In this case, the separation algorithm can be modified to solve this problem by simply adding the connectivity constraints to the set cover model.

4.2.4 Violated Constraint Search Methods

Section 3.2 introduces two approaches for the detection of violated constraints. The strategy taken in the general approach, Algorithm 4.1, is to find violated constraints by directly evaluating of the fort neighborhood finding integer program presented in Section 3.2. Instead however, the random heuristic could be used to search for violated constraints up to a certain number of iterations, and then the integer program could be solved if no violated constraints

were identified. Overall, a combined approach like this is likely to quickly identify violated constraints if they exist, while still being able to verify solutions are optimal if no violated constraints exist. One possible scheme is presented as Algorithm 4.3:

Algorithm 4.3 Mixed Heuristic Separation Algorithm

1: $\mathscr{M} \leftarrow \mathscr{M}_0$
2: Add set cover constraints for sets in $\mathscr{M}_0$ to Model 4.1
3: $S \leftarrow$ optimal solution for LP relaxation of Model 4.1
4: **while** Random Heuristic with weights S returns a solution M **do**
5: Add set cover constraint for set M to Model 4.1
6: $S \leftarrow$ optimal solution for LP relaxation of Model 4.1
7: Set vertex weights for Model 3.1 to S
8: **if** IP 3.1 is feasible **then**
9: $M \leftarrow$ optimal solution for Model 3.1
10: Add set cover constraint for set M to Model 4.1
11: **go to** 4
12: **if** S is fractional **then**
13: Branch on most fractional S_v, **go to** 4
14: **return** S

Chapter 5

Computational Results

In this chapter, I compare the runtime performance of the power dominating infection model presented by Brimkov, Mikesell, and Smith [10] with the models presented in this thesis. In their work, the authors compare their model with that of other known computational methods for the power dominating set problem, and demonstrate improved performance of their algorithm over the other variants of power domination infection models in power domination literature. In order to ensure a fair comparison, I have implemented all algorithms on the same software platforms and have run all computational experiments using the same hardware. Experiments presented in this chapter were implemented using Python 3.5 and Gurobi 7.5.2, and ran on a laptop with 16GB of RAM and an i7 2.80GHz processor.

The graphs used to compare the various methods are predominantly standard IEEE Bus graphs, which are commonly used in electrical engineering literature as electrical test cases for the PMU Placement Problem as well as in power domination literature for comparing computational methods. The graphs considered here are available for download at the University of Washington Power Systems Test Archive [17].

In the first section of this chapter, I present the runtime performance of the unmodified infection model given by Brimkov, Mikesell, and Smith [10] to create a baseline. Next, I show that the performance of their proposed model can be further improved by including the requirement that feasible solutions consist of degree three or higher vertices. Since each test graph considered is connected and contains a vertex of degree 3 or more, by Corollary 2.1 an optimal solution of this form is guaranteed to exist in each test case. Experiments

in this section show that in addition to finding optimal solutions, under this requirement runtime performance is consistently improved.

In the latter section, I present the computational results of the separation algorithm developed in this thesis.

5.1 Power Domination Infection Model

In this section, the runtime performance for Integer Program 1.4, the method introduced by Brimkov, Mikesell, and Smith [10], is evaluated in both its original form and with the added constraint that power dominating sets only include vertices of degree 3 or more. The unmodified model is reproduced in Section 1.5. The modified version is given as Integer Program 5.1.

Integer Program 5.1. Modified Power Domination Infection Model:

$$\text{Min:} \quad \sum v \in V s_v$$

$$\begin{aligned}
\text{s.t.} \quad & s_v + \textstyle\sum_{e \to v} y_e = 1 && \forall v \in V && (1)\\
& x_u - x_v + (T+1)y_e \leq T && \forall e = (u,v) \in E && (2)\\
& x_w - x_v + (T+1)y_e \leq T + (T+1) && \forall e = (u,v) \in E, \forall w \in N(u) \backslash \{v\} && (3)\\
& s_v = 0 && \forall v \in V : \deg(v) \leq 2 && (4)\\
& x_v \in \{0,1,\ldots,T\}, && \forall v \in V, &&\\
& s_v, y_e \in \{0,1\} && \forall e \in E &&
\end{aligned}$$

Power Domination Infection Model Computational Experiments						
Graph				Run Time (s)		
Name	$\|V\|$	$\|J(G)\|$	$\gamma_P(G)$	Inf. Model	Inf. Model*	Speed Up
IEEE Bus 14	14	7	2	0.0446	0.0340	$\times$ 1.33
IEEE Bus 30	30	12	3	0.124	0.103	$\times$ 1.20
IEEE Bus 57	57	24	3	0.653	0.489	$\times$ 1.34
RTS-96	96	43	6	3.306	1.811	$\times$ 1.83
IEEE Bus 118	118	55	8	6.475	5.471	$\times$ 1.18
IEEE Bus 300	300	155	30	97.088	46.138	$\times$ 2.10

Figure 5.1 : Computational experiments comparing the performance of Integer Program 1.4 (Inf. Model) and modified version Integer Program 5.1 (Inf. Model*) demonstrate a modest, yet consistent, improvement for runtime performance.

The experiments shown in Figure 5.1 compare the performance of the unmodified infection model with the performance of the modified infection model. Notably, the improvement is significantly larger for some test cases. The magnitude of this improvement is likely to be dependant upon the structure of specific graphs. Previously, restrictions of this nature have not been considered for the power dominating set problem. It is possible that other solutions of special forms may be guaranteed to exist or nearly always exist, and provide more significant performance improvements when only these solutions are considered.

In the case of power dominating sets of only degree 3 or more vertices, the possible values that the s vector can take is greatly reduced in these test cases. Additionally, these

test sets are considered to be reasonable representations of electrical networks. Thus, the improvement seen in these test cases is expected to extend to other graphs representing electrical networks as well as graphs with similar general structure.

5.2 Power Domination Set Cover Model

In this section, computational experiments regarding the separation algorithm introduced in Chapter 4 are conveyed. Details on how this approach was implemented are given in Section 4.2.

Power Domination Separation Algorithm Computational Experiments						
Graph				Run Time (s)		
Name	$\|V\|$	$\|J(G)\|$	$\gamma_P(G)$	Sep. Alg.	Sep. Alg.*	Speed Up
IEEE Bus 14	14	7	2	0.0101	0.00845	$\times$ 1.19
IEEE Bus 30	30	12	3	0.0169	0.0193	$\times$ 0.88
IEEE Bus 57	57	24	3	0.0807	0.0676	$\times$ 1.19
RTS-96	96	43	6	0.295	0.351	$\times$ 0.84
IEEE Bus 118	118	55	8	1.198	0.757	$\times$ 1.58
IEEE Bus 300	300	155	30	195.520	16.415	$\times$ 11.91

Figure 5.2 : This table details computational experiments comparing the performance of Algorithm 4.1 (Sep. Alg.) and the modified version that uses Integer Program 4.2 as its set cover model (Sep. Alg.*). Here, runtime performance is mainly comparable between the two models, with the exception of the IEEE Bus 300 case. This test case appears to greatly benefit from the additional cuts, as a similarly drastic improvement for this test case was also demonstrated in Figure 5.1.

Interestingly, a major speed up is seen for the Separation Algorithm if the vertex degree constraint is included in test case IEEE Bus 300. In Figure 5.1 a significant speed up is also noted, though not of the same magnitude. This is likely the result of the structure of IEEE 300, and this additional constraint being particularly effective. One avenue for future research is the investigation of graph structures where this additional constraint is particularly effective, and to relate these findings to other previously studied graph parameters.

Power Domination Separation Algorithm Computational Experiments						
Graph				Run Time (s)		
Name	$\|V\|$	$\|J(G)\|$	$\gamma_P(G)$	Inf. Model	Sep. Alg.*	Speed Up
IEEE Bus 14	14	7	2	0.0446	0.00845	$\times$ 5.28
IEEE Bus 30	30	12	3	0.124	0.0193	$\times$ 6.42
IEEE Bus 57	57	24	3	0.653	0.0676	$\times$ 9.69
RTS-96	96	43	6	3.306	0.351	$\times$ 9.42
IEEE Bus 118	118	55	8	6.475	0.757	$\times$ 8.55
IEEE Bus 300	300	155	30	97.088	16.415	$\times$ 5.91

Figure 5.3 : This table directly compares the computational performance of the fully modified version of Algorithm 4.1 (Sep. Alg.*) and the state of the art infection model. Here, a runtime improvement of 5 to 10 times can be seen across all test cases.

Finally, Figure 5.3 compares the ultimate version of the computational methods presented in this thesis with the state of the art infection model. Here, the separation algorithm is considered with the additional cuts provided by Corollary 2.1 as incorporated in Integer Program 4.2. A runtime improvement on all test cases can be observed ranging from approximately 5 to 10 times.

Chapter 6

Conclusions and Future Work

In this final chapter, I give a summary of what I consider to be the key results presented in this thesis, and a discussion of several open problems.

6.1 Summary of Results

In Chapter 2, the Junction Vertex Partition is introduced. For a graph G, this partition establishes a set of vertices of degree three or more vertices called junctions, denoted as $J(G)$, and set of path inducing subgraphs, denoted as $\mathscr{P}(G)$. It is then shown that for any closed neighborhood of a fort, members of the sets $J(G), \mathscr{P}(G)$ are either disjoint from or entirely contained in that neighborhood. Thus, this partition separates the vertices of the graph into building blocks which can be combined to form the neighborhoods of forts. Additionally, it is verified that if G is connected and has $J(G) \neq \varnothing$, then a minimum power dominating set S exists such that $S \subseteq J(G)$.

In Chapter 3, necessary and sufficient conditions for a set $S \subseteq V(G)$ to be the neighborhood of a fort in G are given. Additionally, it is shown that all minimal fort neighborhoods induce connected subgraphs. The task of detecting minimum weight fort neighborhoods is then discussed and a related complexity result is given. Two approaches for finding fort neighborhoods of minimized weight are then given, including an Integer Program whose solutions correspond to fort neighborhoods and a random heuristic.

Next in Chapter 4, a separation algorithm for the minimum power dominating set prob-

lem is given. In this chapter, details regarding the algorithm's implementation are discussed and options for adapting this algorithm for related power domination problems are shared. Finally in Chapter 5, computational experiments comparing the methods presented in this thesis and previous computational methods are presented. These results demonstrate the separation algorithm introduced in Chapter 4 generally outperforms the current state of the art infection model introduced by Brimkov, Mikesell, and Smith [10], with an improvement of $\times 5$ to $\times 10$ in runtime performance.

6.2 Future Work

6.2.1 Subproblem Complexity

At the conclusion of this thesis, several interesting open problems remain. Foremost of these may be the tractability of the Min Weight Fort and Min Weight Fort Neighborhood problems. This thesis proves that these problems are both NP-Complete in general if even a single vertex is known to be in the solution set. In the case of the fort neighborhood problem, it can also be seen that finding a fort neighborhood that contains a given vertex and excludes a given vertex is also an NP-Complete. However, it may be the case that the non-restricted version can be solved with a polynomial complexity algorithm.

This is the case for a well known problem in graph theory called the Even Hole Detection Problem. The Even Hole Detection Problem can be stated as follows: Does graph G contain set C such that $G[C]$ is a cycle and $|C|$ is even? It was shown by Bienstock [6], that this problem is NP-Complete in the case that C must contain a specific vertex. However, a 73-page polynomial time algorithm with complexity of around $O(n^{40}$* was given by Conforti,

*In their paper *Even-hole-free graphs part II: Recognition algorithm*, the authors estimate the complexity of their algorithm to be $O(n^{40})$, but do not formally show it.

Cornuéjols, Kapoor, and Vušković [18], solving the problem in the case that no specific vertex was required to be in the set C. Since the work of Conforti, Cornuéjols, Kapoor, and Vušković, several improvements have been introduced. The algorithm with the best known computational complexity has complexity $O(n^{11})$ and was introduced by Chang and Lu [16].

The Min Weight Fort Neighborhood problem may also permit a polynomial time algorithm for finding fort neighborhoods of weight less than k. It may also be possible to reduce this problem to the Even Hole Detection Problem given the similarity in their computational complexities in the case a vertex is assumed to be in their solutions. It seems likely, however, that even if a polynomial algorithm exists for the min weight fort neighborhood problem it would not perform well in practice.

6.2.2 Minimum Power Dominating Sets of Special Forms

Corollary 2.1 shows that for the vast majority of interesting electrical networks, there is a minimum power dominating set that is entirely contained in $J(G)$. Computational experiments in Chapter 5 show that creating algorithms to specifically look for solutions of this special form can reduce the time needed to find a minimum power dominating set.

If other optimal solutions of special forms exist in all graphs or graphs resembling electrical networks, a similar approach may be devised. As was the case with the special solutions guaranteed by Corollary 2.1, knowing properties of an optimal solution structure may allow a dimension reduction for the problem or the allow for the set of feasible solutions to be significantly reduced.

6.2.3 Kernelization with Junction Vertex Partition

Another promising avenue of future research is the possibility of introducing a kernelization procedure based on the Junction Vertex Partition. As shown in Chapter 2, for any fort neighborhood, every junction path in the graph is entirely contained or disjoint from that neighborhood. This is due to the second part of the power domination color changing rule; if the end of a junction path is colored at any step, then each vertex in that junction path can be colored at a later step. This structure appears ripe for kernelization. These sets are also conveniently identified by the Junction Vertex Partition and junction paths can be cleanly separated into distinct categories.

Bibliography

[1] Ashkan Aazami. Domination in graphs with bounded propagation: algorithms, formulations and hardness results. *Journal of combinatorial optimization*, 19(4):429–456, 2010.

[2] David Amos, Yair Caro, Randy Davila, and Ryan Pepper. Upper bounds on the k-forcing number of a graph. *Discrete Applied Mathematics*, 181:1–10, 2015.

[3] Egon Balas and Shu Ming Ng. On the set covering polytope I: All the facets with coefficients in $\{0, 1, 2\}$. *Mathematical Programming*, 43(1-3):57–69, 1989.

[4] Francesco Barioli, Wayne Barrett, Shaun M. Fallat, H. Tracy Hall, Leslie Hogben, Bryan Shader, P. van den Driessche, and Hein van der Holst. Zero forcing parameters and minimum rank problems. *Linear Algebra and its Applications*, 433(2):401 – 411, 2010.

[5] Katherine F. Benson, Daniela Ferrero, Mary Flagg, Veronika Furst, Leslie Hogben, Violeta Vasilevska, and Brian Wissman. Power domination and zero forcing. *arXiv:1510.02421*, 2015.

[6] Dan Bienstock. On the complexity of testing for odd holes and induced odd paths. *Discrete Mathematics*, 90(1):85 – 92, 1991.

[7] John Adrian Bondy, Uppaluri Siva Ramachandra Murty, et al. *Graph theory with applications*, volume 290. Citeseer, 1976.

[8] Chassidy Bozeman, Boris Brimkov, Craig Erickson, Daniela Ferrero, Mary Flagg, and Leslie Hogben. Restricted power domination and zero forcing problems. *arXiv preprint arXiv:1711.05190*, 2017.

[9] Boris Brimkov, Caleb C. Fast, and Illya V. Hicks. Computational approaches for zero forcing and related problems. *arXiv:1704.02065*, 2017.

[10] Boris Brimkov, Derek M. Mikesell, and Logan A. Smith. Connected power domination in graphs. *arXiv:1712.02388*, 2017.

[11] Dennis J. Brueni and Lenwood S. Heath. The PMU placement problem. *SIAM Journal on Discrete Mathematics*, 19(3):744–761, 2005.

[12] Daniel Burgarth and Vittorio Giovannetti. Full control by locally induced relaxation. *Physical review letters*, 99(10), 2007.

[13] Daniel Burgarth and Vittorio Giovannetti. Full control by locally induced relaxation. *Physical review letters*, 99(10):100501, 2007.

[14] Steve Butler, Laura DeLoss, Jason Grout, H. Tracy Hall, Joshua LaGrange, Tracy McKay, Jason Smith, and Geoff Tims. Minimum rank library, 2014.

[15] Gerard Jennhwa Chang, Paul Dorbec, Mickael Montassier, and André Raspaud. Generalized power domination of graphs. *Discrete Applied Mathematics*, 160(12):1691–1698, 2012.

[16] Hsien-Chih Chang and Hsueh-I Lu. A faster algorithm to recognize even-hole-free graphs. *Journal of Combinatorial Theory, Series B*, 113:141 – 161, 2015.

[17] Rich Christie. Power systems test case archive. *Electrical Engineering dept., University of Washington*, 2000.

[18] Michele Conforti, Gérard Cornuéjols, Ajai Kapoor, and Kristina Vušković. Even-hole-free graphs part II: Recognition algorithm. *Journal of graph theory*, 40(4):238–266, 2002.

[19] William J. Cook, William H. Cunningham, William R. Pulleyblank, and Alexander Schrijver. *Combinatorial Optimization*, volume 33. John Wiley & Sons, 2011.

[20] George Dantzig, Ray Fulkerson, and Selmer Johnson. Solution of a large-scale traveling-salesman problem. *Journal of the operations research society of America*, 2(4):393–410, 1954.

[21] Nathaniel Dean, Alexandra Ilic, Ignacio Ramirez, Jian Shen, and Kevin Tian. On the power dominating sets of hypercubes. In *Computational Science and Engineering (CSE), 2011 IEEE 14th International Conference on*, pages 488–491. IEEE, 2011.

[22] Ruilong Deng, Gaoxi Xiao, and Rongxing Lu. Defending against false data injection attacks on power system state estimation. *IEEE Transactions on Industrial Informatics*, 13(1):198–207, 2017.

[23] Michael Dorfling and Michael A. Henning. A note on power domination in grid graphs. *Discrete Applied Mathematics*, 154(6):1023 – 1027, 2006.

[24] Paul A. Dreyer Jr and Fred S. Roberts. Irreversible k-threshold processes: Graph-theoretical threshold models of the spread of disease and of opinion. *Discrete Applied Mathematics*, 157(7):1615–1627, 2009.

[25] Christina J. Edholm, Leslie Hogben, Joshua LaGrange, and Darren D. Row. Vertex and edge spread of zero forcing number, maximum nullity, and minimum rank of a graph. *Linear Algebra and its Applications*, 436(12):4352–4372, 2012.

[26] Linda Eroh, Cong X. Kang, and Eunjeong Yi. A comparison between the metric dimension and zero forcing number of trees and unicyclic graphs. *Acta Mathematica Sinica, English Series*, 33(6):731–747, 2017.

[27] Neng Fan and Jean-Paul Watson. Solving the connected dominating set problem and power dominating set problem by integer programming. In *International Conference on Combinatorial Optimization and Applications*, pages 371–383. Springer, 2012.

[28] Caleb C. Fast. *Novel Techniques for the Zero-Forcing and p-Median Graph Location Problems*. PhD thesis, Rice University, 2017.

[29] Michael Gentner, Lucia D. Penso, Dieter Rautenbach, and Uéverton S Souza. Extremal values and bounds for the zero forcing number. *Discrete Applied Mathematics*, 214:196–200, 2016.

[30] Mohammad Ghamsari-Yazdel and Masoud Esmaili. Reliability-based probabilistic optimal joint placement of PMUs and flow measurements. *International Journal of Electrical Power & Energy Systems*, 78:857–863, 2016.

[31] Martin Grötschel, László Lovász, and Alexander Schrijver. The ellipsoid method and its consequences in combinatorial optimization. *Combinatorica*, 1(2):169–197, 1981.

[32] Martin Grötschel, László Lovász, and Alexander Schrijver. *Geometric algorithms and combinatorial optimization*, volume 2. Springer Science & Business Media, 2012.

[33] Teresa W. Haynes, Sandra M. Hedetniemi, Stephen T. Hedetniemi, and Michael A Henning. Domination in graphs applied to electric power networks. *SIAM Journal on Discrete Mathematics*, 15(4):519–529, 2002.

[34] Leslie Hogben, Nicole Kingsley, Sarah Meyer, Shanise Walker, and Michael Young. Propagation time for zero forcing on a graph. *Discrete Applied Mathematics*, 160(13):1994–2005, 2012.

[35] Gaoqi Liang, Steven R. Weller, Junhua Zhao, Fengji Luo, and Zhao Yang Dong. The 2015 ukraine blackout: Implications for false data injection attacks. *IEEE Transactions on Power Systems*, 32(4):3317–3318, 2017.

[36] Chung-Shou Liao and Der-Tsai Lee. Power domination problem in graphs. In *International Computing and Combinatorics Conference*, pages 818–828. Springer, 2005.

[37] Chung-Shou Liao and Der-Tsai Lee. Power domination in circular-arc graphs. *Algorithmica*, 65(2):443–466, 2013.

[38] Yao Liu, Peng Ning, and Michael K. Reiter. False data injection attacks against state estimation in electric power grids. *ACM Transactions on Information and System Security (TISSEC)*, 14(1):13, 2011.

[39] Nikolaos M. Manousakis, George N. Korres, and Pavlos S. Georgilakis. Taxonomy of PMU placement methodologies. *IEEE Transactions on Power Systems*, 27(2):1070–1077, 2012.

[40] Anamitra Pal, Anil Kumar S. Vullikanti, and S. S. Ravi. A PMU placement scheme considering realistic costs and modern trends in relaying. *IEEE Transactions on Power Systems*, 32(1):552–561, 2017.

[41] Simone Severini. Nondiscriminatory propagation on trees. *Journal of Physics A: Mathematical and Theoretical*, 41(48), 2008.

[42] Ranjana. Sodhi, S. C. Srivastava, and Sri Niwas Singh. Multi-criteria decision-making approach for multi-stage optimal placement of phasor measurement units. *IET Generation, Transmission & Distribution*, 5(2):181–190, 2011.

[43] Xin Tai, Damián Marelli, Eduardo Rohr, and Minyue Fu. Optimal PMU placement for power system state estimation with random component outages. *International Journal of Electrical Power & Energy Systems*, 51:35–42, 2013.

[44] AIM Minimum Rank-Special Graphs Work et al. Zero forcing sets and the minimum rank of graphs. *Linear Algebra and its Applications*, 428(7):1628–1648, 2008.

[45] Guangjun Xu, Liying Kang, Erfang Shan, and Min Zhao. Power domination in block graphs. *Theoretical Computer Science*, 359(1-3):299–305, 2006.

[46] Min Zhao, Liying Kang, and Gerard J. Chang. Power domination in graphs. *Discrete mathematics*, 306(15):1812–1816, 2006.