SOLUTION TECHNIQUES FOR LARGE-SCALE OPTIMIZATION PROBLEMS ON THE TRANSMISSION GRID

A Dissertation
Presented to
The Academic Faculty

By

Emma S. Johnson

In Partial Fulfillment
of the Requirements for the Degree
Doctor of Philosophy in the
College of Engineering
H. Milton Stewart School of Industrial and Systems Engineering

Georgia Institute of Technology

December 2021

SOLUTION TECHNIQUES FOR LARGE-SCALE OPTIMIZATION PROBLEMS ON THE TRANSMISSION GRID

Thesis committee:

Dr. Santanu Dey
Industrial and Systems Engineering
Georgia Institute of Technology

Dr. Daniel Molzahn
Electrical and Computer Engineering
Georgia Institute of Technology

Dr. Anya Castillo
Principal Research Scientist
NextEra Analytics

Dr. Andy Sun
Industrial and Systems Engineering
Georgia Institute of Technology

Dr. Mathieu Dahan
Industrial and Systems Engineering
Georgia Institute of Technology

Date approved: November 5, 2021

ACKNOWLEDGMENTS

First of all, thank you to my advisor, Santanu Dey, for becoming my advisor under difficult circumstances, weathering uncountably many arguments, and most of all, being supportive, gentle, and motivating through everything (including a pandemic). Thank you also to my committee members: Your ideas have improved this thesis greatly, (and, in the chapter on transmission switching, added to the mystery).

I would also like to thank my many colleagues and mentors through the years at Sandia National Laboratories: John Siirola, this is all your fault, and I couldn't be more grateful for your support, advice, and your faith in my intuition. Bryan Arguello, your friendship, willingness to help, and creativity have saved me many a time. Anya Castillo, thank you for always setting the bar sky-high and then having unwavering confidence that I could meet it. Cindy Phillips, from technical advice to personal advice, your help has been invaluable and our conversations some of the most fun in my week. Thank you also to Sean DeRosa, Jonathan Eckstein, Jared Gearhart, Bill Hart, Brian Pierre, and Jean-Paul Watson for all the time, ideas, and help you have given me.

Last, the support of my family and friends means the world to me. To my parents Laurie and Curtis Johnson, it could be said that you overdid it instilling curiosity in me, but it has paid off! Kirthana Hampapur, from the moment another early bird walked into the first-year PhD offices, I have had the best friend of my life. Thank you Paritosh Ramanan for your friendship and your spirited discussions about research, Alexandre Velloso for your friendship, perspective, and determination in our collaborations, Mohamed El Tonbari for being the best lab-mate imaginable, Jana Boerger for being ready to brainstorm about anything, and Kaitlyn Garifi for understanding me perfectly and demonstrating to me that PhDs are possible. And thank you Erik Wijmans for your friendship, speculations about k-nearest neighbors, and the thousands of hours we have spent rock climbing in the past five years–a principal reason my sanity remains intact.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ACRONYMS

ACOPF AC Optimal Power Flow

DCOPF DC Optimal Power Flow

DCOTS DC Optimal Transmission Switching

ISO Independent System Operator

KKT Karush-Kuhn-Tucker

KNN k-nearest neighbors

MILP mixed-integer linear programming

MIP mixed-integer programming

NFLB network flow lower bound

PTDF Power Transfer Distribution Factor

SCADA Supervisory Control and Data Acquisition

NOTATION

Sets

$\mathcal{L}$ Set of transmission lines

$\mathcal{G}$ Set of generators

$\mathcal{S}$ Set of buses, also called substations

$\mathcal{R}$ Set of relays

$\mathcal{B}$ Set of balancing authorities

$\mathcal{C}$ Set of control centers

$\mathcal{N}$ Set of nodes in communication network: $\mathcal{N} = \mathcal{B} \cup \mathcal{C} \cup \mathcal{S}$

$\mathcal{G}_s$ Set of generators at bus $s \in \mathcal{S}$

$\mathcal{L}_s^+$ Set of lines to bus $s \in \mathcal{S}$

$\mathcal{L}_s^-$ Set of lines originating at bus $s \in \mathcal{S}$

$\mathcal{R}_l$ Set of relays which control line $l \in \mathcal{L}$

$\mathcal{R}_g$ Set of relays which control generator $g \in \mathcal{G}$

$\mathcal{R}_s$ Set of relays which control bus $s \in \mathcal{S}$

$\overline{K}$ Set of attacks on the communication network (defined by the sets of relays they compromise)

$E(n)$ Set of possible enclaves in node $n \in \mathcal{N}$

$C(n)$ Set of child nodes of node $n \in \mathcal{N}$

$R(n)$ Set of relays that are descendants of node $n \in \mathcal{N}$. Note that $R(n) = C(n)$ for $n \in \mathcal{S}$

$A(k)$ Relays compromised in attack $k \in \overline{K}$

Parameters

$o(l)$ Origin bus of transmission line $l \in \mathcal{L}$

$d(l)$ Destination bus of transmission line $l \in \mathcal{L}$

$s(g)$ Bus where generator $g \in \mathcal{G}$ is located

B_l Susceptance of transmission line $l \in \mathcal{L}$

$\overline{F}_l$ Thermal limit for transmission line $l \in \mathcal{L}$

$\overline{P}_g$ Upper limit of generator g dispatch level

$\underline{P}_g$ Lower limit of generator g dispatch level

D_s Demand at bus $s \in \mathcal{S}$

C_g Per-unit generation cost of generator $g \in \mathcal{G}$

$P(n)$ Parent of node $n \in \mathcal{N}$

R Infeasibility cost for the DCOTS problem

K Maximum number of lines that can be opened

T Maximum number of balancing authority and control center enclaves network designer can add to the control network

U Maximum number of relays that the attacker can compromise

V Maximum number of substation enclaves network designer can add to the control network

W Maximum number of enclaves that the attacker can compromise

Variables

In the following description, a variable defined as "whether or not" some statement holds is binary, being 1 if the statement is true and 0 if it is false.

f_l Power flow through transmission line l

p_g Generator dispatch level for generator g

ℓ_s Load shed at bus s

o_s Over-generation at bus s

θ_s Phase angle for bus s

y_l Indicator of whether or not line l is closed

δ_r Indicator of whether or not relay r is compromised

u_s Whether or not load at substation s is compromised, that is, all the load is shed

v_l Whether or not transmission line l is compromised, that is, its flow is forced to zero

w_g Whether or not generator g is compromised, that is, its power generation is forced to zero.

λ_l^+ Dual of line flow upper bound

λ_l^- Dual of line flow lower bound

μ_s Dual of power balance constraint

φ_s Dual of load shed lower bound

ψ_s Dual of load shed upper bound

γ_g Dual of generator dispatch upper bound

$x_{n,i}$ Whether or not enclave i in node n is active (added to the network)

$y_{n,i,j}$ Whether or not enclave j in node $P(n)$ is the parent of enclave i in node n (not to be confused with y_l, which only appears in Chapter 2)

$q_{r,j}$ Whether or not relay r is controlled by enclave j in control center $P(P(r))$

$z_{s,i}$ Whether or not any relay belonging to substation s is a child of enclave i in control center $P(s)$.

$\alpha_{n,i}$ Whether or not enclave i in node n is compromised

$\beta_{s,i}$ Whether or not to compromise the enclave in substation s that is controlled by enclave i in control center $P(s)$

$t_{n,i,k}$ Whether or not enclave i in node n has to be compromised to execute attack k

$\tau_{s,i,k}$ Whether or not any relay controlled by a substation s which is controlled by enclave i in its parent control center must be compromised to execute attack k.

SUMMARY

In this thesis, we are interested in solution techniques and primal heuristics for several large-scale optimization problems on the transmission grid. While some of these problems have been studied for a long time, none of the techniques proposed previously allowed them to be solved exactly on large-scale networks, rendering them of little use in practice. We will present methodology which yields high quality solutions on large networks.

In Chapter 2, we consider the DC Optimal Transmission Switching (DCOTS) problem. In this problem, we simultaneously optimize the grid topology (i.e., choose which lines are on and off) and the generator dispatch, using a DC Optimal Power Flow (DCOPF) model. It is well-known that transmission switching is an affordable way to reduce congestion in the grid, reducing generation costs. However, DCOTS has so far been a prohibitively difficult model to solve, particularly given that dispatch problems are solved every 5 to 10 minutes for most Independent System Operators (ISOs). We present a data-driven approach which assumes that DCOTS has been solved to optimality offline for a variety of demand profiles. We then use the k-nearest neighbors (KNN) method as a primal heuristic, directly mapping from demand profiles to topologies. This scales well since the computational time is dominated by the time to solve k linear programs. We find that we can generate high-quality primal solutions within the time constraints imposed by real-time operations. In addition, we find that defining the feature space for KNN differently can also yield equally good results: In particular, using dual information from the DCOPF problem can be effective.

In the Chapter 3, we propose a scalable lower bound for a worst-case attack on transmission grid relays. This is a bilevel interdiction problem in which an attacker first targets relays within an attack budget, compromising all components controlled by the relays he chooses, and aiming to maximize load shed. Then, a defender redispatches the generators, solving a DCOPF model and minimizing the load shed. Since the inner problem is convex,

it is possible to dualize it, resulting in a mixed-integer single-level reformulation. However, the difficulty arises in linearizing this reformulation. Without bounds on the dual variables of DCOPF, this is not possible to do exactly. Prior literature has used heuristic bounds on the duals. However, in addition to providing a lower bound only, this comes at a computational cost: The more conservatively the bounds are chosen, the larger the big-M values are in the resulting mixed-integer programming (MIP) formulation. This worsens the continuous relaxation and makes it increasingly difficult for even commercial solvers. Instead, we propose using a different lower bound: We relax DCOPF to capacitated network flow, dropping the constraints corresponding to Ohm's law. We show that, on uncongested networks, the injections we get from solving this relaxation are a good approximation of those from solving DCOPF. We can again dualize this problem, creating a single-level formulation. The duals of the relaxed defender problem are bounded in absolute value by 1, meaning we can linearize the single-level formulation and solve it exactly. Furthermore this MIP scales extremely well when solving with a commercial solver.

Last, in Chapter 4, we present methodology to solve a trilevel interdiction problem where the inner bilevel problem is the worst-case relay attack problem from Chapter 3. In this problem, we optimize the design of the Supervisory Control and Data Acquisition (SCADA) network controlling the transmission grid in order to minimize the impact of a worst-case cyberattack. Specifically, we decide where the cyber networks in the SCADA system should be subdivided, with communication limited between these subdivisions, a technique called *network segmentation*. The resulting problem is a trilevel interdiction model with pure integer first and second player problems and a convex third-player problem. We show that it can be solved for large-scale power networks using a covering decomposition approach in which we iteratively fix a network design and generate a worst-case attack. We then find a new design that makes all the generated attacks infeasible. When there is no such design, then the design corresponding to the least-damaging attack gen-

erated so far is optimal. We show empirically that this method is scalable for large power networks and moderate network designer and attacker budgets.

CHAPTER 1
INTRODUCTION

The power grid is well-known to be critical to much of modern life. However, it is vulnerable to random equipment failures, natural disasters, and intentional physical and cyber-attacks, among other threats. In the United States, as of 2013, over 70% of the transmission lines and transformers were over 25 years old (Executive Office of the President Council of Economic Advisers, 2013). This means both that this equipment is vulnerable to failure or weather damage and that it predates much of the cyber infrastructure used today to control the transmission grid. Severe weather is the most common cause of power outages (Ton and W. P. Wang, 2015), but cyberattacks such as the one on the Ukrainian Grid in 2015 and the spring 2021 Colonial Pipeline disruption underscore the vulnerability of SCADA systems to intentional attacks as well as the far-reaching consequences of these attacks. Furthermore, the percentage of electricity generation from renewable sources in the United States has increased to 21% in 2020, and is projected to continue increasing by nearly 10% per year (U.S. Energy Information Administration, 2021). These are likely not the only upcoming changes as the United States strives to reach the goal of carbon pollution free electricity by 2050 (Horowitz, 2021). It follows that, as grid infrastructure, the threats it faces, and society's dependence on it evolve, so too should our ability to operate it efficiently, invest in it wisely, and anticipate threats to it before they are realized.

Optimization problems for the transmission grid range from long-term planning problems to enhance grid resiliency, security, and efficiency to medium- and short-term operational problems such as a scheduling generator commitment, maintenance, and dispatch. A common element among many of these problems is that they involve decisions which change the grid's topology. In the case of investment planning this might be in the form of adding new equipment or reconfiguring existing infrastructure (both cyber and physical). In

the case of operational problems, this could come in the form of deciding which generators and lines are on when or deciding how storage systems are utilized. While conceptually simple, all of these problems are inherently combinatorial and are usually NP-hard. As a result, there is ongoing need for methodology to find good solutions to such problems at realistic scales.

Unsurprisingly, at the heart of all the previously mentioned optimization problems is the need to model power flow. In reality, power flow is governed by physical laws, namely Ohm's Law and Kirchhoff's Law. It needs to also obey physical limits on the equipment such as thermal limits on the lines. The resulting formulation is referred to as AC Optimal Power Flow (ACOPF) and leads to non-convex quadratic optimization problems. Particularly for large-scale problems, using the ACOPF formulation is intractable, so it is common to use a convex relaxation in its place. Kocuk, Dey, et al. (2016) give a detailed discussion of ACOPF and several convex relaxations of it. The simplest relaxation, called DCOPF, is a linear relaxation which relies on assuming that the voltage angle differences between buses will be small, that voltage magnitudes are known, and that the line resistances are significantly less than the line reactances. These assumptions usually hold in typical operations of the transmission grid, though Nagarajan et al. (2017) and Garifi et al. (2021) show DC models have limitations for applications such as modeling grid response in extreme weather conditions.

In the following chapters, we will examine several problems which are still difficult to solve even using the DCOPF approximation, namely optimal transmission switching and finding worst-case contingencies where a fixed number of grid components are compromised, referred to as $N - k$ *contingencies*. While both of these problems have been modeled using ACOPF (see for example Kocuk, Dey, et al. (2016) for AC optimal transmission switching and Sundar, Coffrin, et al. (2018) for a comparison of power flow relaxations for the $N - k$ problem), we will show in the following chapters that for large-scale power networks, even the DCOPF versions of these problems are computationally challenging using

conventional methodology. For this reason, the focus of the thesis will be on techniques for solving these versions at scale, and, in the case of the $N - k$ problem, we will show that we can further simplify the power flow model to capacitated network flow while still maintaining good solution quality.

Specifically, in Chapter 2, we present a KNN heuristic for the DCOTS problem which relies on historical data for the network. In Chapter 3, we consider a variation of the $N - k$ problem where a cyberattacker gains access to a budgeted number of the relays which control the transmission grid components, seeking to maximize the resulting loss of load. We refer to this as the *worst-case relay attack problem.* Last, in Chapter 4, we consider a trilevel model which optimizes the decisions of a network designer who can modify the cybernetworks which control the transmission grid in order to minimize the loss of load from the same worst-case attack we consider in Chapter 3. We refer to this as the *network segmentation problem.*

All of the problems we consider in this thesis have to do with changing the grid topology for various reasons: In optimal transmission switching, the topology is adjusted in order to minimize generation costs. In contrast, in the worst-case relay attack problem and in the network segmentation problem, an attacker attempts to change the topology (as well as the power injections) in order to maximize unserved load. While the goals differ, the means do not. In particular, in all of these problems, the optimal solutions will exploit transmission line congestion in the grid (or a lack thereof). For example, in the absence of congestion, transmission switching is unnecessary. In the KNN heuristic, we are essentially using the distance between two instances of the problem in parameter space as a proxy for how similar the two instances are in terms of where they cause congestion in the network. In the worst-case relay attack problem, the difficulty is again because of congestion: The attacker can sometimes do more damage than just the sum of the capacities of the components he compromises because he can exploit congested areas of the network to waste generation capacity. However, we show that in this problem, we need not even model DCOPF, and we

can instead simplify to capacitated network flow. This method exploits a lack of congestion, taking advantage of the fact that thermal limits on lines are usually much higher than is necessary in typical operations. Last, in the network segmentation problem, we again take advantage of a lack of congestion by using the network flow restriction.

Changing the grid topology is not straightforward since it will change where in the network the lines are congested, that is, where they are operating at or near their thermal limits in order to move the power as needed. As we will explore in more detail at the end of Chapter 2, where a network becomes congested is difficult to understand. It is a function both of the original structure and the demands at a given point in time. In optimal transmission switching, there is a need to both anticipate areas of congestion and to know good topologies which mitigate that congestion. These questions are integrally related to what the power flow on each line is at a given time. In contrast, in both the worst-case relay attack problem and the network segmentation problem, we are making not an operational decision, but a planning decision (or analysis). In this case, we are interested not in the line flows, but in the injections, that is, what load is served and shed, and what amount of power the generators are outputting. It is for this reason that we are able to abstract power flow even further, relying on capacitated network flow to approximate the injections necessary to make the planning decisions. Thus, while many applications still do need more accurate power flow models such as convex relaxations of the ACOPF equations, in this thesis we show several problems which are already computationally demanding with even the DCOPF approximation, and, in the case of the problems in Chapters 3 and 4, may require less accurate modeling of power flows.

CHAPTER 2

A K-NEAREST NEIGHBORS HEURISTIC FOR REAL-TIME DC OPTIMAL TRANSMISSION SWITCHING

While transmission switching is known to reduce power generation costs, the difficulty of solving even DC Optimal Transmission Switching (DCOTS) has prevented optimal transmission switching from becoming commonplace in real-time power systems operation. In this paper, we present a k-nearest neighbors (KNN) heuristic for DCOTS which relies on the insight that, for routine operations on a fixed network, the DCOTS solutions for similar load profiles and generation cost profiles will likely turn off similar sets of lines. We take a data-driven approach and assume that we have DCOTS solutions for many historical instances, which is realistic given that the problem is solved every 5 minutes in practice. Given a new instance, we find a set of "close" instances from the past and return the best of their solutions for the new instance. We present a case study on 7 test networks ranging in size from 118 to 3,375 buses. We compare the proposed heuristic to DCOTS heuristics from the literature, commercial solver heuristics, and a simple greedy local search algorithm. In most cases, we find better quality solutions in less computational time. In addition, the computational time is within the limits imposed by real-time operations, even on larger networks. We then present an empirical study of our training data to understand why the heuristic works well. Last, we show some variants of the KNN heuristic which also work well.

2.1 Introduction

Transmission switching is an inexpensive way to reduce generation costs in a congested power system. Ideally, we want to always dispatch the cheapest generators first, which we call a merit-order dispatch. However, network constraints such as line flow limits and

Ohm's law can make this dispatch infeasible. Due to Braess' paradox, by not allowing flow through certain lines, we can reduce network congestion and approach the desired merit-order dispatch (Blumsack et al., 2007). Fisher et al. (2008) show that an optimal network topology for one load profile is not necessarily optimal for another, presenting a need for real-time switching as load fluctuates.

Transmission switching in real-time poses a challenge since the DC Optimal Power Flow (DCOPF) problem is solved every 5 minutes by most Independent System Operators (ISOs) to determine generator dispatch for the next time interval. This means that the DCOPF problem itself should be solved in significantly less than 5 minutes to allow time in the remainder of the 5-minute interval for feasibility checks and other post-processing. Since DCOTS requires co-optimizing the generator dispatch and the network topology, to be useful in practice, DCOTS needs to be solved in less than 5 minutes. This is computationally challenging since DCOTS is NP-hard, and also cannot be approximated within a constant factor (Lehmann et al., 2014; Kocuk, Jeon, et al., 2016).

Various MIP formulations and heuristics for DCOTS have been proposed, but none scale well enough that they could be implemented in real-time. There are a variety of formulations in the literature, but, because they scale poorly with network size, they can at best be used as heuristics by limiting the set of candidate lines for switching. Kocuk, Jeon, et al. (2016) propose a cycle-based formulation for DCOTS and use it to derive valid inequalities leveraged in a cutting plane solution approach. The cuts improve computational time, but the approach does not scale such that it could be used in real-time. Ruiz, Goldis, et al. (2017) propose a smaller formulation for security-constrained DCOTS based on the Power Transfer Distribution Factor (PTDF) power flow equations. By security-constrained, we mean the solution is also feasible under a set of predefined contingency scenarios. This formulation can be used as a heuristic since it is computationally efficient when the number of contingencies is small and the set of candidate lines for switching is limited. Jabarnejad (2018) proposes an approximate model for DCOTS which is guaranteed to yield feasible

solutions with the same generation costs but lower numbers of switched lines. However, using this approximation for large networks still entails solving a large mixed integer program, which does not yet scale well.

The difficulty of MIP formulations of DCOTS has so far prohibited exactly optimizing the topology on large networks within the time limit imposed by real-time. However, numerous heuristics have been proposed. Ruiz, Foster, et al. (2012) use four different metrics based on sensitivity analysis of DCOPF in order to estimate the cost benefit of switching each line of the network. They use these estimates to prioritize the lines. They then iterate through this priority list, switching off lines that result in cost savings. Similarly, Fuller et al. (2012) use one of the same sensitivity criteria to choose a subset of high priority lines. They then run a greedy algorithm over just this set of lines, switching off the line resulting in the most cost savings until there are no more cost-saving lines or until they reach a maximum cardinality of switched lines. The sensitivity-based heuristics are effective in terms of solution quality. However, the number of linear programs they solve scales with the product of the number of lines in the system and the budget of how many lines to switch. This makes them impractical for large networks and for switching large numbers of lines.

Others have proposed using sensitivity information or historical data to limit the set of switchable lines so as to reduce the number of binaries in the DCOTS problem, e.g. Liu et al. (2012). In an effort to understand the economic impacts of switching via a case study on a congested network, Hedman et al. (2011) suggest two heuristics for DCOTS. One involves a variation of the greedy algorithm used in Fuller et al. (2012), but uses a partitioning technique to explore multiple disjoint sets of feasible solutions in parallel. The second uses data from past instances to select a limited set of switchable lines which were cost-beneficial to switch off in the past. They then solve the DCOTS problem allowing only these lines to be opened.

The idea of using data from past solves has recently begun to take hold in the form of learning-based heuristics. Since the dispatch problem is solved so frequently, and under

normal conditions demands and generation costs for a given time period do not vary beyond around 10% and 5% respectively (Xavier et al., 2020), a likely avenue for scalable DCOTS heuristics is to harness off-line computational power and develop a heuristic based on the solutions to historical instances of the problem.

There has been recent interest in applying machine learning methods to power systems problems. Xavier et al. (2020) present three learning-based methods which use historical data to solve the security constrained unit commitment problem. Yang and Oren (2019) apply KNN, an artificial neural network, and decision tree regression to learn sets of high-priority lines to consider for switching. They then use a greedy algorithm based on Fuller et al. (2012), which uses this line prioritization to generate a topology. In addition, they use machine learning methods to train an algorithm selection oracle which, given an instance, chooses which among these algorithms to run.

We propose a KNN approach different than that of Yang and Oren (2019). We apply the method for learning initial feasible solutions from Xavier et al. (2020) to DCOTS. Rather than training an oracle to map parameter vectors to sets of high priority lines, as in Yang and Oren (2019), we instead use k-nearest neighbors to learn topologies directly from the instance data. More specifically, we assume that we have a large collection of solved instances of DCOTS. Given a new instance, our heuristic selects the nearest k instances in parameter space out of this solved collection. We then test the quality of each of the optimal topologies of the k nearest instances and return the lowest-cost topology as our switching solution. We show through case studies on seven different transmission networks ranging from 118 to 3,375 buses that we achieve, on average, better solutions than other heuristics in less time, making our heuristic potentially practical for real-time operations. Even for larger networks, the number of solved training instances needed is moderate. A collection of 270 training instances yields solutions that are usually within 2% of the best known solution and, in most cases, are within 1% of the best known solution. Another advantage of the method is that it scales well for larger networks, never taking more than 40 seconds

on any of our test instances. This is largely because we never iterate through lists of lines. Instead, the effect of the size of the network is on training time, which is done offline, and on the computational time for the linear program (LP) solves which check the cost of the near topologies. However, this effect is minimal since there are only k such solves.

In the remainder of the chapter, we give an overview of our heuristic approach in Section 2.2, details on the networks used in our case study in Section 2.3, computational results benchmarking our heuristic against others from the literature in Section 2.4, an analysis to give some intuition behind the algorithm in Section 2.5, some potential variations on the algorithm in Section 2.6, and we conclude in Section 2.7.

2.2 Heuristic Approaches

The DCOTS problem is given formally in Appendix A. In this section, we will first present our proposed heuristic in Section 2.2.1. We then present five heuristics that will be used as benchmarks.

2.2.1 KNN Heuristic

Our proposed heuristic relies on the assumption that the DCOTS problem has likely been solved on the same network many times. We assume the variable data are the load profiles and the generation costs. All other parameters are known, and are constant on a given network. We can therefore characterize an instance of DCOTS as the vector of generation costs appended to the vector of demands. If q is such a vector, we will denote an instance of DCOTS as $I(q)$. Suppose we have a collection of solved DCOTS instances $\mathcal{Q}$, each with an ϵ-optimal transmission switching solution. Given a new instance $I(q)$, we propose a heuristic based on KNN to find a transmission switching solution: Among the set of solved instances, we find the nearest k instances as measured by their parameter vectors' distances from q using a p-norm for some p. We then test the transmission switching solutions from each of these k closest instances. We return the transmission switching solution which has

the best objective value. The algorithm is presented formally in Algorithm 4 in Appendix B.

Note that the solution returned by Algorithm 4 will respect the cardinality constraint since all the solutions to the training instances do. However, the solution is not guaranteed to be feasible, which is why we have included slacks on the nodal balance constraints in the DCOTS formulation given in Appendix A. Since we are already using the DC approximation for power flow, the solution would already have to be corrected for feasibility in practice, so we allow these small violations.

The scalability of this algorithm relies mainly on the value of k and the number of training instances in $\mathcal{Q}$. Thus, though the size of the network can dramatically increase the training time, the only effect it has on the algorithm's computational time is to increase the time spent in the k DCOPF solves in the loop beginning at line 13 of Algorithm 4. The time taken in the loop beginning at line 3 depends on $|\mathcal{Q}|$. In our experiments, we found that $|\mathcal{Q}| = 270$ was sufficient. We tested Algorithm 4 with both Euclidean and ℓ_∞-norms.

2.2.2 Greedy Local Search

We compare the above heuristic to the four heuristics from Ruiz, Foster, et al. (2012), to a greedy local search algorithm, presented as the line enumeration algorithm in Yang and Oren (2019), and to Gurobi's primal heuristics. For completeness, we describe the greedy algorithm here. We first calculate the cost, fixing all lines closed. Then, for each line, we fix only that line open and calculate the cost. If none of the lines improve the cost when opened, we are done. Else, we fix open the line that improves the cost the most. We repeat this process on the remaining set of lines that could be opened, terminating either when we see no improvement from any of the lines or when we have opened as many lines as the cardinality constraint will allow. The algorithm is given formally in Algorithm 5 in Appendix B.

Note that this algorithm scales poorly since it requires solving nearly $K \cdot |\mathcal{L}|$ linear programs. Also note that, in a very congested system, where the solution with all lines closed is not feasible, this algorithm also does not guarantee a feasible solution, and the heuristics from Ruiz, Foster, et al. (2012) do not either.

2.2.3 Sensitivity-Based Heuristics

We also compare to the four sensitivity-criteria-based heuristics from Ruiz, Foster, et al. (2012). For brevity, we do not describe the algorithms in detail here, but note that each of these heuristics uses sensitivity information from the DCOPF problem in order to calculate criteria to indicate lines which are likely to be cost-beneficial to open. Each heuristic uses a different such criterion to order the lines for consideration in a greedy algorithm. That is, we follow the procedure from Algorithm 5, but the set S is ordered based on the criterion. The four criteria are the Line Profits criterion, the Price Difference criterion, the Total Cost criterion, and the PTDF-Weighted Cost criterion. Note that, even without a cardinality restriction, the four different criteria can find different solutions since each switching decision in the greedy algorithm is conditioned on the ones before it, so the ordering of S does change the heuristic solution.

2.2.4 Gurobi Heuristics

Last, we compare to Gurobi's performance when we run it with the `Heuristics` parameter set to 1, meaning that it spends all its time on primal heuristics. We also warmstart these runs with the solution where all lines are closed, which is an obvious feasible solution in all but very congested cases.

2.3 Test Cases

We test the heuristic on the 118-bus test system as modified in Blumsack (2006), on the 300-bus test system as modified in Kocuk, Jeon, et al. (2016), on the 1354 and 2869 bus

Table 2.1: Test instance sizes

Test Case	Number of Buses	Number of Generators	Number of Lines
118blumsack	118	19	186
300kocuk	300	61	411
1354pegase	1,354	260	1,991
1951rte_api	1,951	391	2,596
2869pegase	2,869	510	4,582
2869pegase_api	2,869	510	4,582
3375wp_api	3,374	596	4,161

PEGASE systems (Fliscounakis et al., 2013), and on modified versions of the 1951 RTE, 2869 PEGASE, and 3375 Polish system from Babaeinejadsarookolaee et al. (2019). For these last three instances we used the heavily loaded versions of the systems, which we have labeled with the postfix "api," as in the Babaeinejadsarookolaee et al. (2019) library. All but the 118-bus and 300-bus systems were downloaded from the IEEE PES Power Grid Library (Babaeinejadsarookolaee et al., 2019): The heavily loaded versions are from v20.07 and the 1354 and 2869 PEGASE systems are from v19.05. The instances have varying levels of congestion, but we chose only instances which have a cost benefit from transmission switching. For example, the original version of the 1951 RTE case was not selected because, with the nominal demands and generation costs, there is no benefit to opening lines. Details of these instances are shown in Table 2.1.

For each of these systems, we generate 300 instances following the methodology from Xavier et al. (2020). For completeness, we describe the process here: Suppose that d^0 is the original vector of demands and c^0 is the original vector of generation costs. For $i \in \{1, 2, \ldots, 300\}$, for each bus $s \in \mathcal{S}$, draw β_s^i from a uniform distribution on the interval $[0.9, 1.1]$. Then let $d_s^i = \beta_s^i d_s^0$. This process results in 300 demand profiles $d^1, d^2, \ldots, d^{300}$. We generate the generation costs in exactly the same way except that we only allow 5% variation around the nominal generation cost, so we draw from a uniform distribution on $[0.95, 1.05]$. We solve all 300 of the instances to 1%-optimality or to a time limit of 0.5 hour, whichever comes first. We randomly select 30 instances of these 300 as test instances

and leave the other 270 for training. In Appendix C, using these test and training sets, we give further detail regarding the congestion of these test instances and their potential cost benefit from transmission switching.

2.4 Computational Results

We report results running the KNN heuristic with $k = 10$ and using both the ℓ_2-norm and the ℓ_∞-norm to measure the distance from the training instances. We give our heuristic and the Gurobi heuristics a time limit of 5 minutes. We give the four heuristics from Ruiz, Foster, et al. (2012) a time limit of 10 minutes to forgive inefficiencies in our implementation. For all the heuristics, we test for cardinality limits of 5 and 10, and also with no cardinality constraint. For the 118blumsack case, we also compare to the greedy local search algorithm given in Section 2.2.2, but we find it to be the worst-performing heuristic in this case and intractable for larger test instances.

Our model and all our heuristics are implemented in Pyomo (Bynum et al., 2021; Hart et al., 2011), relying on Gurobi version 9.0.2 as the solver (Gurobi Optimization, LLC, 2020). We solve on a server with 96 Intel Xeon 2.30GHz processors and 529GB RAM. For all our experiments, we constrain Gurobi to 8 threads.

Our results for the 30 test instances are shown in the boxplots in Figures 2.1 through 2.7. In all the plots, the middle line is the median value of the statistic shown, the bottom and top of the box are the 1st and 3rd quartiles respectively, and the "whiskers" extend to the largest (in absolute value) value from the box which is no further than 1.5 times the interquartile range. Outliers beyond that range are shown as individual points.

The plots in the left-hand column of each figure show the distributions of solution quality for each of the heuristics, where we measure solution quality as the relative gap of the objective from the best-known solution for the test instance. That is, the gap is given by $\frac{z-\hat{z}}{\hat{z}}$, where z is the cost of the heuristic solution and $\hat{z}$ is the cost of the best known solution for the instance. Note that, since we only solved the training instances to 1%-optimality

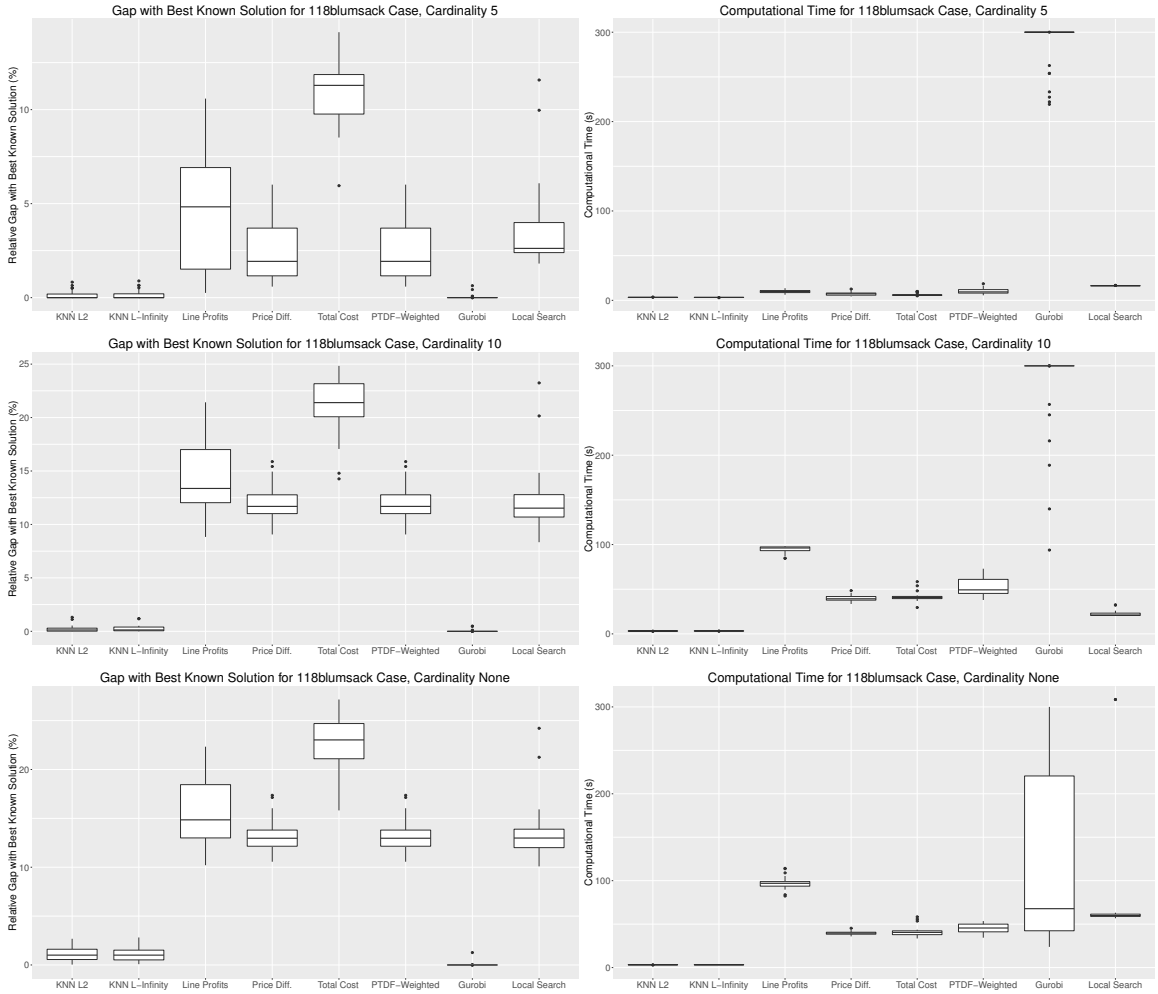


Figure 2.1: Solution quality and computational time results for the 118blumsack case for the three different cardinality options.

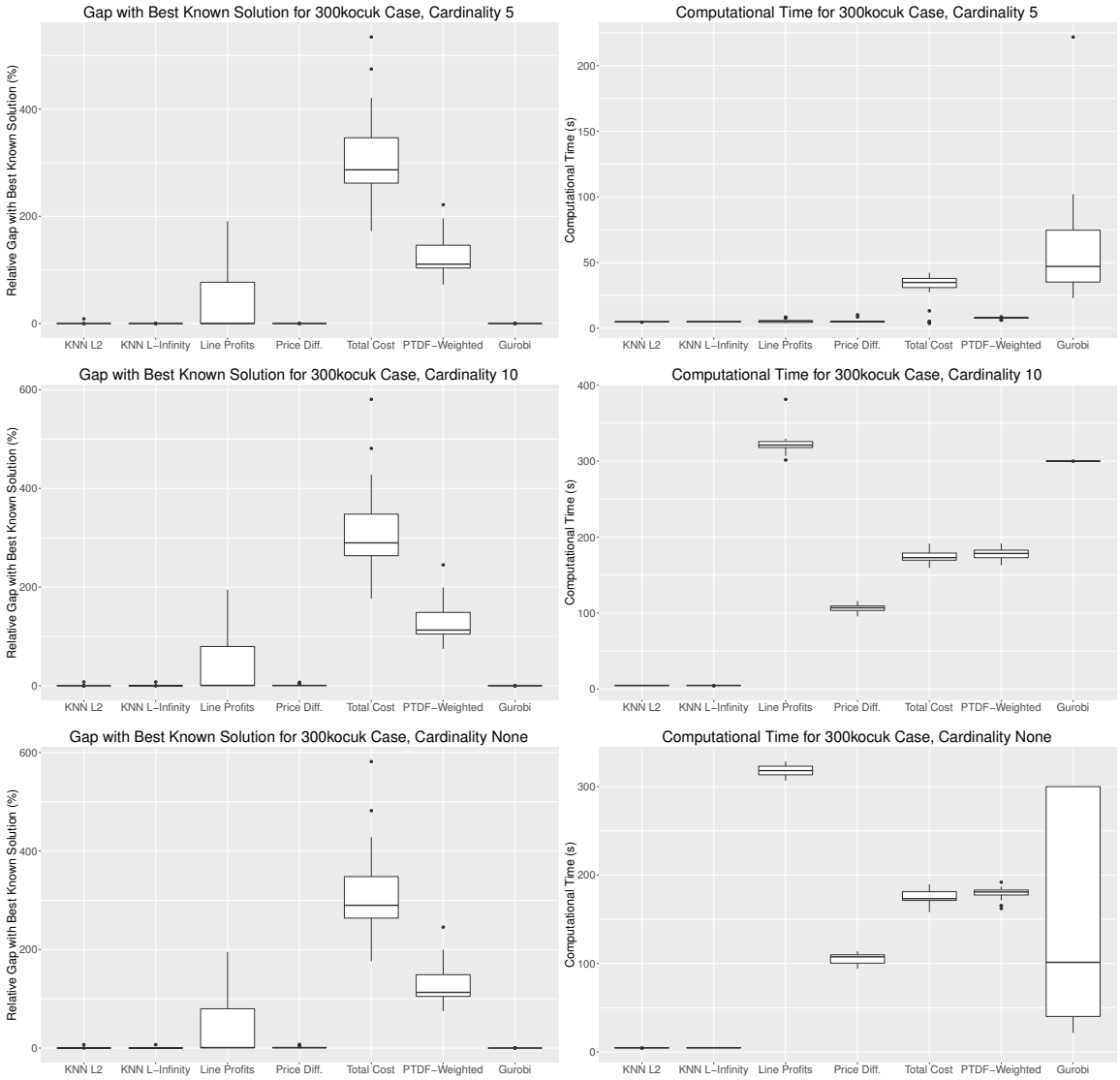


Figure 2.2: Solution quality and computational time results for the 300kocuk case for the three different cardinality options.

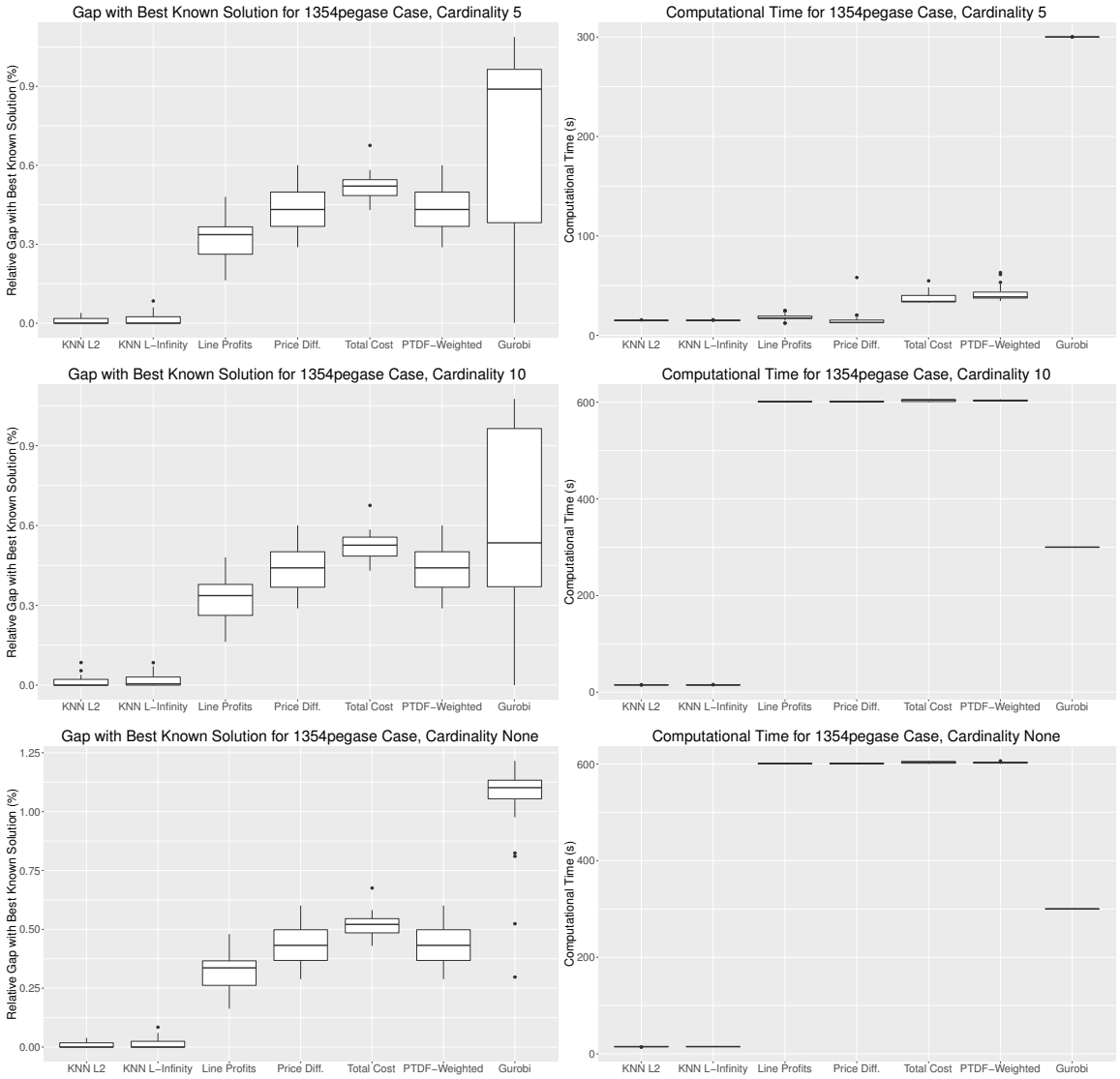


Figure 2.3: Solution quality and computational time results for the 1354pegase case for the three different cardinality options.

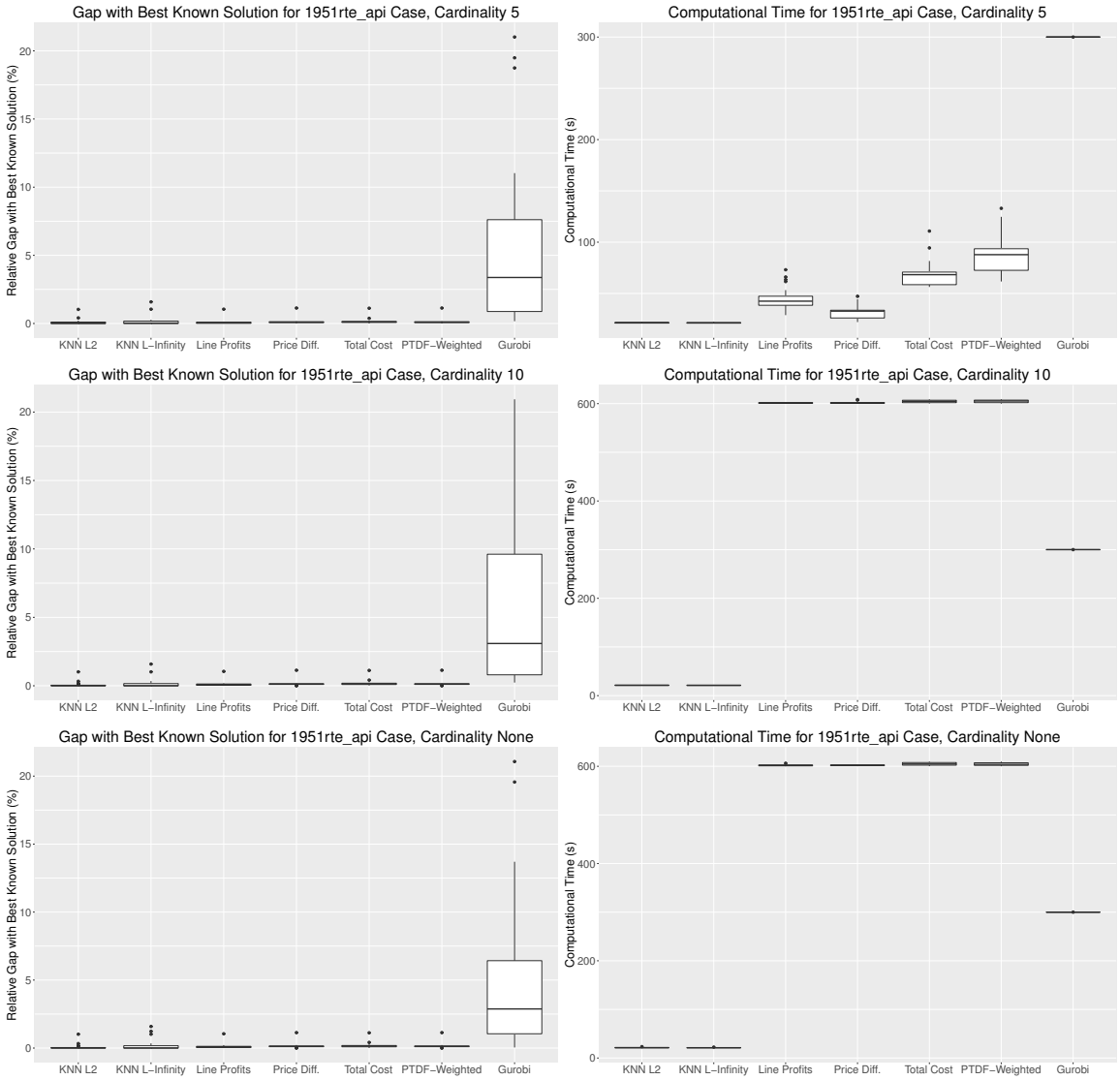


Figure 2.4: Solution quality and computational time results for the 1951rte_api case for the three different cardinality options.

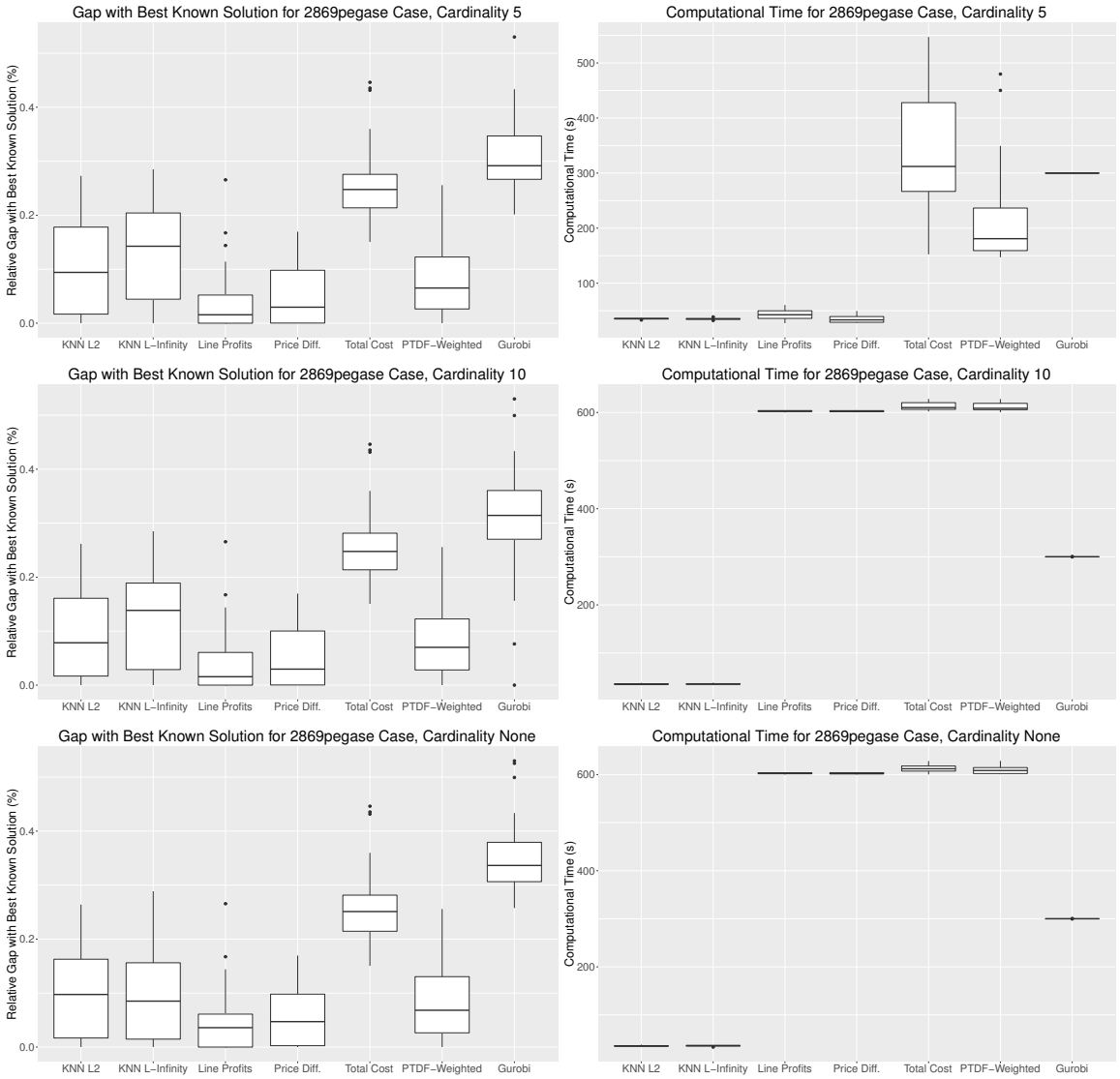


Figure 2.5: Solution quality and computational time results for the 2869pegase case for the three different cardinality options.

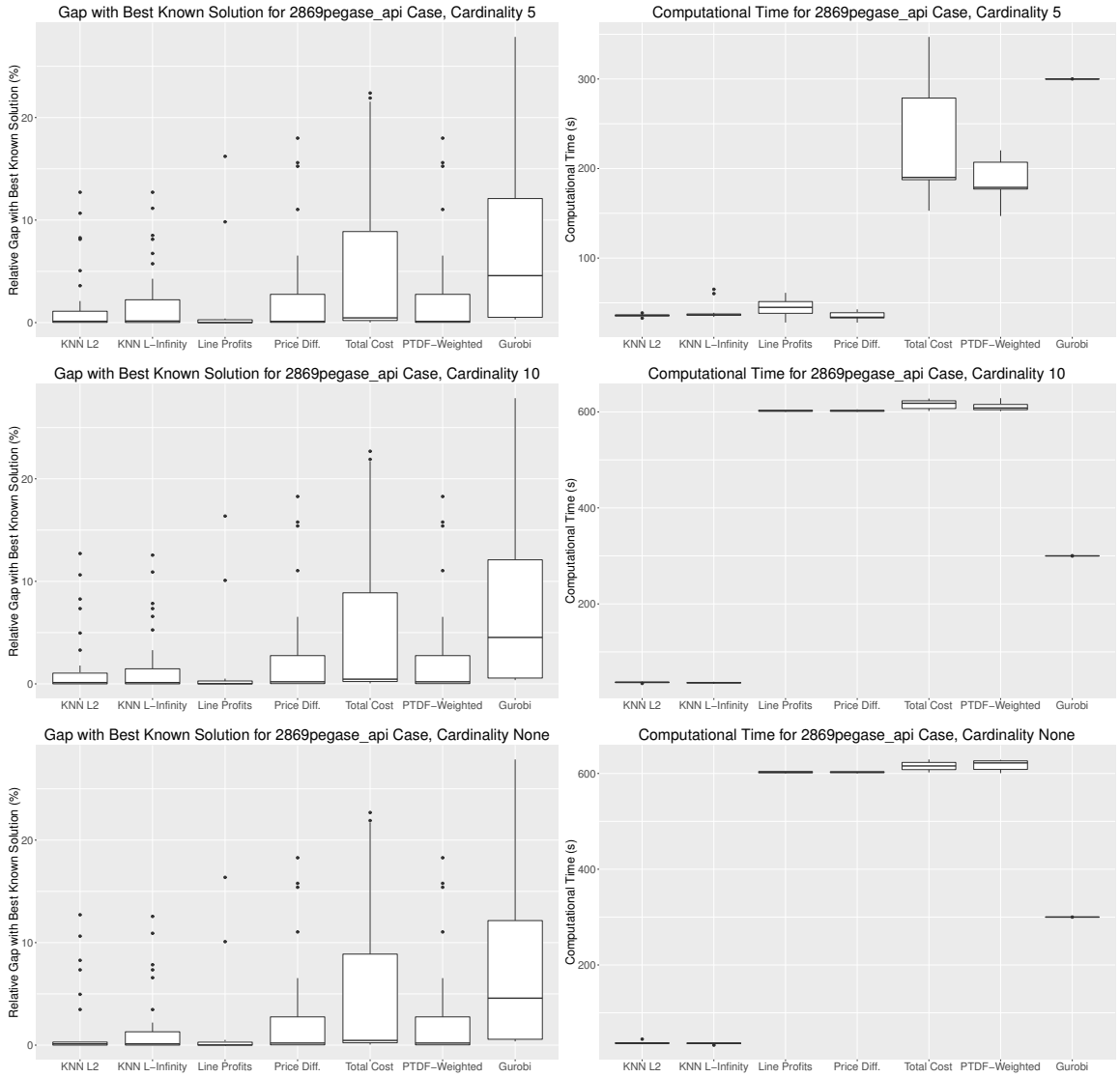


Figure 2.6: Solution quality and computational time results for the 2869pegase_api case for the three different cardinality options.

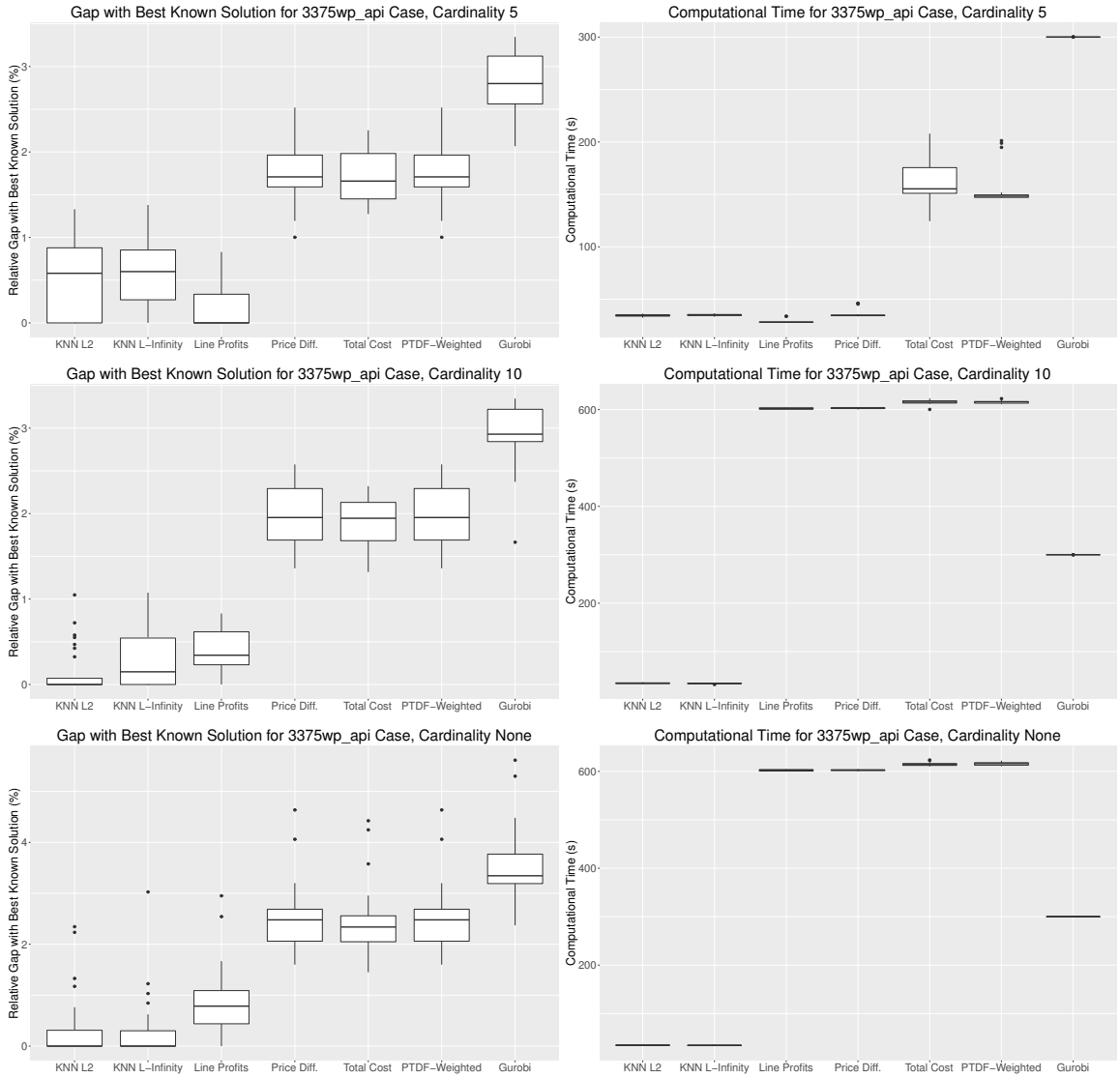


Figure 2.7: Solution quality and computational time results for the 3375wp_api case for the three different cardinality options.

and we stopped at a time limit, it is possible to find a new best-known solution via any of the tested heuristics. Thus, our best-known solution is not necessarily the solution we calculated during training: It is the best known solution across training and all of the presented heuristics.

The right-hand column of plots shows the distributions of computational times for each of the heuristics. Note that, for the Total Cost and PTDF-Weighted Cost heuristics, we do not include the time needed to calculate the Load Outage Distribution Factor (LODF) matrix since that only needs to be done once for a network and so can be treated as data.

2.4.1 Computational Time

In all of our test cases and for all the variations on the cardinality constraint, the KNN heuristic has one of the least computational times. Even on the 3375wp_api case, it runs for 35 seconds on average, and always takes less than 40 seconds. This is because the time is completely independent of the cardinality of the set of opened lines and is also relatively agnostic to the size of the network. Regardless of these, the time to calculate the solution from the KNN heuristic is the time to find the k closest instances from among the 270 training instances, and the time to solve the k DCOPF problems fixing the solutions from these k instances. Thus, the time only scales up slowly with network size, since network size increases the DCOPF solve times slightly. In contrast, for the sensitivity-based heuristics and for the greedy local search algorithm, the time scales up with increased cardinality and with the number of lines in the system. The Gurobi heuristics also do not appear to scale well for increases in the cardinality constraint right-hand side and for increased network size.

In several cases, the Line Profits and Price Difference sensitivity-based heuristics have comparable computational times in the cardinality 5 variant. However, even in the 118blumsack case, these heuristics are no longer competitive for a cardinality of 10. Gurobi is never competitive in terms of computational time, often reaching its 5-minute time limit. The

Table 2.2: Number of test instances for which each heuristic finds the best solution out of all the heuristic solutions. Ties are counted as both of the heuristics finding the best solution, explaining why the sums of the rows can exceed 30.

Test Case	Cardinality	KNN L2	KNN L-Infinity	Line Profits	Price Diff.	Total Cost	PTDF-Weighted	Gurobi
118blumsack	5	4	3	0	0	0	0	26
	10	3	3	0	0	0	0	26
	None	0	0	0	0	0	0	30
300kocuk	5	10	10	0	1	0	0	19
	10	11	9	0	0	0	0	19
	None	2	1	0	0	0	0	28
1354pegase	5	22	20	0	0	0	0	0
	10	22	17	0	0	0	0	3
	None	22	20	0	0	0	0	0
1951rte_api	5	12	15	8	0	1	0	0
	10	17	16	4	0	0	0	0
	None	18	16	4	0	0	0	0
2869pegase	5	5	2	10	8	0	7	0
	10	5	2	10	8	0	6	1
	None	5	4	9	7	0	7	0
2869pegase_api	5	6	6	19	0	0	0	0
	10	8	8	16	0	0	0	0
	None	8	8	16	0	0	0	0
3375wp_api	5	9	4	19	0	0	0	0
	10	22	12	5	0	0	0	0
	None	20	17	2	0	0	0	0
Total Over All Test Cases	5	68	60	56	9	1	7	45
	10	88	67	35	8	0	6	49
	None	75	66	31	7	0	7	58
Total		231	193	122	24	1	20	152

Local Search algorithm was only slightly worse than the KNN heuristic and the sensitivity-based heuristics for the 118blumsack case, though its computational time increases with cardinality, like the sensitivity-based heuristics.

2.4.2 Solution Quality

In Table 2.2, we show, for each test case, cardinality variant, and heuristic, the number of times that the heuristic finds the best solution out of all the heuristics in our experiments. (The Local Search algorithm is omitted since it never finds the best solution.) The second-to-last three rows of Table 2.2 show the total number of times each heuristic found the best solution across all the test cases, but still subdivided by cardinality. In the last row, we

show the overall totals, that is, the sum of the previous three rows. Both the ℓ_2-norm and ℓ_∞-norm variants of the KNN heuristic find the best solution for more test cases than any of the other heuristics, and the ℓ_2-norm is the best by this measure. When we break these numbers down by cardinality, while the KNN heuristics still find the best solution for more test instances, we see that the margin by which they do so increases as cardinality increases.

In the 118blumsack test case for all cardinalities, Gurobi's heuristics are competitive with the KNN heuristic in terms of solution quality, and usually find a better solution. However, this is at the cost of computational time: The KNN heuristic runs for at most 3.5 seconds whereas Gurobi runs for at least 24 seconds, but usually for 5 minutes. For the 300kocuk case, both the Price Difference heuristic and Gurobi are competitive with the KNN heuristic, but this is again at the cost of time, except in the cardinality 5 version, where the Price Difference heuristic is also very fast. For the 1354pegase case also, the KNN heuristic yields higher quality solutions than all the comparisons, and it does so in less time in all but the cardinality 5 case.

In the 1951rte_api, 2869pegase, 2869pegase_api, and 3375wp_api cases, one or more of the heuristics from Ruiz, Rudkevich, et al. (2012) match the KNN heuristic in terms of solution quality, despite the fact that they are terminated early by the time limit for the cardinality 10 case and the case without a cardinality constraint. In particular the Line Profits and Price Difference heuristics tend to perform well. However, for all of these cases, only the KNN heuristic achieves this quality in less than 5 minutes for the larger cardinalities, and, in the case of 1951rte_api, KNN is even slightly faster for cardinality 5. Last, note that, while some of the sensitivity-based heuristics do sometimes find better solutions than the KNN heuristic, the monetary savings from these are limited. For example, even in the cardinality 5 version of the 2869pegase case, where perhaps this difference is most extreme, the Line Profits heuristic is on average a 0.06% improvement over the ℓ_2-norm version of the KNN heuristic, which translates to an additional \$1,451 in cost savings. While not insignificant, this shows that the difference in solution quality may be of less importance

for less-congested systems where the potential cost savings from transmission switching is limited. In contrast, for the 1354pegase case with cardinality 5, the average savings of the ℓ_2-norm version of the KNN heuristic over the Line Profits heuristic is \$3,700. In summary, how much solution quality matters is partly a function of congestion, since some systems have the potential for much more impact from transmission switching.

In general, we see that the solution quality from the KNN heuristic is fairly constant across different sized networks and different cardinality constraints, usually achieving a relative gap between 0.01% and 1.0%. In contrast, it seems that the performance of the sensitivity-based heuristics is more variable for different power networks. With a limit on switching only 5 lines, the Line Profits and Price Difference heuristics also tend to achieve good quality solutions in very little computational time, but in Table 2.2, we see that overall, across all our test systems and cardinality variants, the KNN heuristics find the best solution more often.

2.4.3 Feasibility

As mentioned in Section 2.2.1, it is possible that the KNN heuristic returns a topology which requires load shed or over-generation. Since using the DC power flow approximation already means that post-processing is required to ensure true feasibility, we allow this to happen. We found that, for the 118blumsack, 1354pegase, and 2869pegase test cases, our heuristic never returned a solution which had a positive value for load shed or over-generation. The other four networks all did have infeasible solutions returned, despite our relatively high value of R. However, this is also the case for the other heuristics: For some of our generated instances, it is very difficult to find a feasible solution or the instance is slightly infeasible, and none of the tested heuristics guarantee feasibility.

In Figure 2.8, we show the distribution of load shed for the test instances for each of the test cases where infeasibility occurs. We show this for the cardinality 5 variant, since for instances that require switching to be feasible, this will be the hardest to find feasibility. The

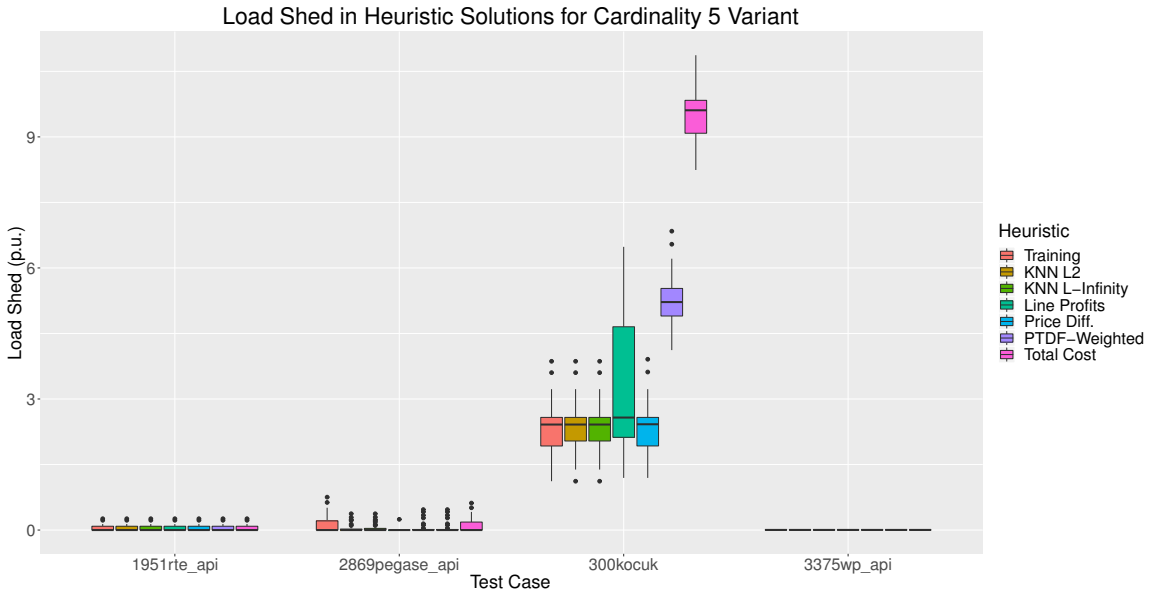


Figure 2.8: The distribution of the load shed (in per unit) for each of the heuristic solutions to the test instances which are ever infeasible.

same plot for the other two cardinality variants looks similar. We also plot the distributions of load shed in the training solutions, which is about the same as those of the heuristics. Our instance generating procedure is not guaranteed to give feasible instances, and we did not filter the instances for feasibility. Some of them are not feasible without transmission switching and some of them are not feasible even with transmission switching allowed. While this would not be a problem in routine operations, transmission switching has been proposed for contingency response and for disaster response, so it is worthwhile to test our heuristic on cases which require load shedding.

Note that, in all cases where the KNN heuristic returned an infeasible solution, all k topologies we tested were infeasible: This means it is not the value of R which was at fault. While it may seem that increasing the value of k could mitigate this issue, we are achieving similar levels of infeasibility as the training set, implying that k is also not the reason for this behavior. In all but the 300kocuk case, the worst load shed for any heuristic solution from the Euclidean norm version of the KNN heuristic is less than 38MW, and on average is less than 5MW. This is actually a slight improvement over the load shed in

the training solutions, which is at worst 76MW and on average no worse than 14MW. In general the KNN heuristics and the heuristics from Ruiz, Foster, et al. (2012) have the same levels of infeasibility, though the Line Profits heuristic has less load shed on average for the 2869pegase_api case. The 300kocuk case is not feasible even in the nominal case, explaining the much higher amounts of load shed for all the heuristic solutions. Lastly, we only saw positive over-generation for two test instances in the 300kocuk case and in two training solutions in the 2869pegase_api case. In all of these solutions, the over-generation was less than 2MW.

2.4.4 Parameter Tuning

The difference between the Euclidean and ℓ_∞-norms is minimal in most of the experiments. Unsurprisingly, the two are essentially indistinguishable in terms of computational time. Overall, the ℓ_∞-norm version of the heuristic has a more variable performance and usually is slightly worse on average. It also tends to have more extreme outliers, occasionally performing much worse than the Euclidean norm on a test instance. It therefore seems that the Euclidean norm is preferable, as it has both a more consistent and slightly improved performance on the test instances.

Lastly, though the results presented here all have 270 training instances and 10 as the value of k, we also experimented with the KNN heuristic's sensitivity to these parameters. Even for the large networks, 270 training instances appears sufficient: There was very little benefit in increasing to 500, though reducing to 100 did hurt the heuristic's performance. We also experimented with setting k to 3 and 5. Both of these perform well, though for smaller k we saw small amounts of load shed for the version of the 118blumsack case without a cardinality constraint. Overall, since the computational time is manageable when k is 10 and the quality and feasibility are improved, the larger value seems preferable.

Table 2.3: Leave-one-out cross validation results: Mean, minimum, and maximum relative gaps of the KNN heuristic solution compared to the best known solution.

Test Case	Cardinality								
	5			10			None		
	mean	min	max	mean	min	max	mean	min	max
118blumsack	0.10%	0.00%	1.48%	0.24%	0.00%	3.06%	0.71%	0.00%	12.00%
300kocuk	0.02%	0.00%	4.49%	0.10%	0.00%	5.24%	0.09%	0.00%	5.23%
1354pegase	0.00%	0.00%	0.07%	0.00%	0.00%	0.07%	0.00%	0.00%	0.07%
1951rte_api	0.06%	0.00%	3.68%	0.06%	0.00%	3.68%	0.06%	0.00%	3.68%
2869pegase	0.01%	0.00%	0.19%	0.01%	0.00%	0.21%	0.01%	0.00%	0.21%
2869pegase_api	0.31%	0.00%	16.9%	0.43%	0.00%	17.4%	0.45%	0.00%	17.4%
3375wp_api	0.03%	0.00%	1.46%	0.03%	0.00%	1.11%	0.08%	0.00%	3.40%

2.4.5 Leave-one-out Cross Validation

We ran leave-one-out cross validation on the 300 instances for each of the cardinalities with $k = 10$ and using the Euclidean norm. Results are shown in Table 2.3. On average, across all of the test systems and cardinalities, the heuristic returns a result within 0.75% of the best-known solution. In addition, the heuristic is consistent: It appears that some instances of the 2869pegase_api network could benefit from a larger value of k, but with the exception of this network, the worst-case is within 6% of the best known solution, and is usually even better.

2.5 Reasons the KNN Heuristic Works

While it is difficult to give theoretical justification for the good performance of the KNN heuristic with the small training set, it is possible to prove that KNN can find the correct integer solution for general MIPs with a sufficiently large training set. Formally, we have the following proposition:

Proposition 1. *Suppose we have the following problem:*

$$\begin{aligned} g(b) := \min_{x,y} \quad & c_x^T x + c_y^T y \\ \textit{s.t.} \quad & Ax + Gy \geq b \\ & x \in \{0,1\}^n \\ & y \in \mathbb{R}^m \end{aligned} \tag{2.1}$$

Let (x^*, y^*) be an optimal solution which is unique in the integer component and let there be a neighborhood of b such that there exists a feasible solution of (2.1) everywhere in this neighborhood, with the integer component being x^*. Then there exists a neighborhood of b such that for all $\hat{b}$ in this neighborhood, x^* is the integer part of the optimal solution to the corresponding problem.

A proof of Proposition 1 is given in Appendix D. Note that, according to Proposition 1, if we had a training set such that the union of corresponding neighborhoods covered the space of all possible testing demands, then the KNN heuristic would find the optimal solution. In practice, this amount of training data is not realistic, but as we saw in Section 2.4, it does not appear necessary. We suspect that the success of the KNN heuristic on DCOTS is due to the properties of the problem. Our arguments about why the KNN heuristic works are:

1. The switching solutions which are optimal within normal variance in demands tend to be composed of similar sets of lines. That is, for a given network, certain lines appear good to open in general. We discuss this further in Section 2.5.1.

2. There are many near-optimal solutions for a given instance of DCOTS, and optimal solutions for close instances in parameter space tend to be near-optimal for their neighbors. We give results related to this in Section 2.5.2.

3. Changes in demand at certain nodes behave nearly identically to changes in demand at other nodes when viewed in terms of their effect on the optimal topology. In essence, changes in demand at individual buses are less important than the overall effect on the congestion patterns in the network. This is discussed further in Section 2.5.3. We find that aggregating certain regions of the network into a net injection leads to no loss in performance for the KNN heuristic.

Points 1 and 3 suggest that, while the KNN heuristic treats the optimal DCOTS solution as a function of the demands and the generation costs, it is really a function of the congestion pattern of the network. Somewhat pathologically, we will show in Section 2.5.2 that, while near solutions in parameter space tend to be high-quality, the best solutions are further. Thus, while distance is a good proxy for clustering instances with the same DCOTS solution, there could be room for improvement in modeling congestion patterns and their effect on transmission switching decisions.

2.5.1 Few Lines are Ever Opened in DCOTS Solutions

In the following, we will present some results analyzing our training sets for the instances presented in Section 2.3. Though there are many unique DCOTS solutions in our training set, the number of lines ever opened in any training solution is small, and furthermore, some sets of lines tend to be opened together. This does not mean, however, that every solution in the training set is near-optimal for our test instances. The fact that the KNN heuristic considers solutions of problems close in the parameter space still appears important to finding high-quality solutions.

In the left section of Table 2.4, we show the number of unique DCOTS solutions in our set of 300 generated instances for each cardinality variant. This number is given as a percentage of 300 in parentheses. Notice that with few exceptions, we tend to have nearly 300 unique DCOTS solutions in this set. For the lower-cardinality variants of 118blumsack

Table 2.4: Number of unique topologies and number of lines which are opened in the 300-instance training sets for different test cases. In each entry, the raw number is shown first, and then in parentheses we give that number as percentage of the 300 test instances in the case of unique solutions and as a percentage of the total number of lines in the system in the case of the number of lines opened.

Test Case	Number of Unique Solutions Cardinality						Number of Lines Opened in Any Solution Cardinality					
	5		10		None		5		10		None	
118blumsack	55	(18.3%)	161	(53.7%)	300	(100.0%)	27	(14.5%)	69	(37.1%)	145	(78.0%)
300kocuk	6	(2.0%)	139	(46.3%)	300	(100.0%)	7	(1.7%)	65	(15.8%)	277	(67.4%)
1354pegase	247	(82.3%)	247	(82.3%)	247	(82.3%)	181	(9.1%)	197	(9.9%)	569	(28.6%)
1951rte_api	280	(93.3%)	298	(99.3%)	300	(100.0%)	158	(3.4%)	220	(4.8%)	948	(20.7%)
2869pegase	283	(94.3%)	284	(94.7%)	292	(97.3%)	226	(3.4%)	299	(6.5%)	1217	(26.6%)
2869pegase_api	294	(98.0%)	299	(99.7%)	300	(100.0%)	240	(5.2%)	352	(7.7%)	2816	(61.5%)
3375wp_api	276	(92.0%)	294	(98.0%)	296	(98.7%)	124	(3.0%)	179	(4.3%)	2689	(64.6%)

and 300kocuk, this is not true, perhaps because these cases are so congested that there are fewer topologies which make sense, despite variations in demand and generation costs.

In the right section of Table 2.4, we show the number of lines which are opened in any of the 300 generated topologies, both as a raw number and as a percentage of the total lines in the system. In general, and especially for the lower-cardinality variants, few lines are ever beneficial to open. In addition, certain lines are almost always beneficial to open, and certain sets of lines are often opened together. For example, in the training set for 118blumsack with cardinality 5, one line is switched off in 78% of the unique solutions. Another is switched off in 72% of the solutions. However, a third line, which is opened in 28% of the solutions, is only opened in combination with the first line mentioned in 16% of the solutions and with the second line mentioned in 12% of the solutions. These patterns suggest that, not only do the same lines get switched in many different solutions, certain sets of lines get switched off together. This is good evidence suggesting that heuristics common in the literature which limit the set of switchable lines are finding high-quality solutions if this set is chosen wisely.

Because certain lines get opened very often in all the training solutions, it seems possible that all of the training solutions are near-optimal. This is, however, not the case. In Figure 2.9, we plot the distribution of relative gaps with the training solution, fixing ev-

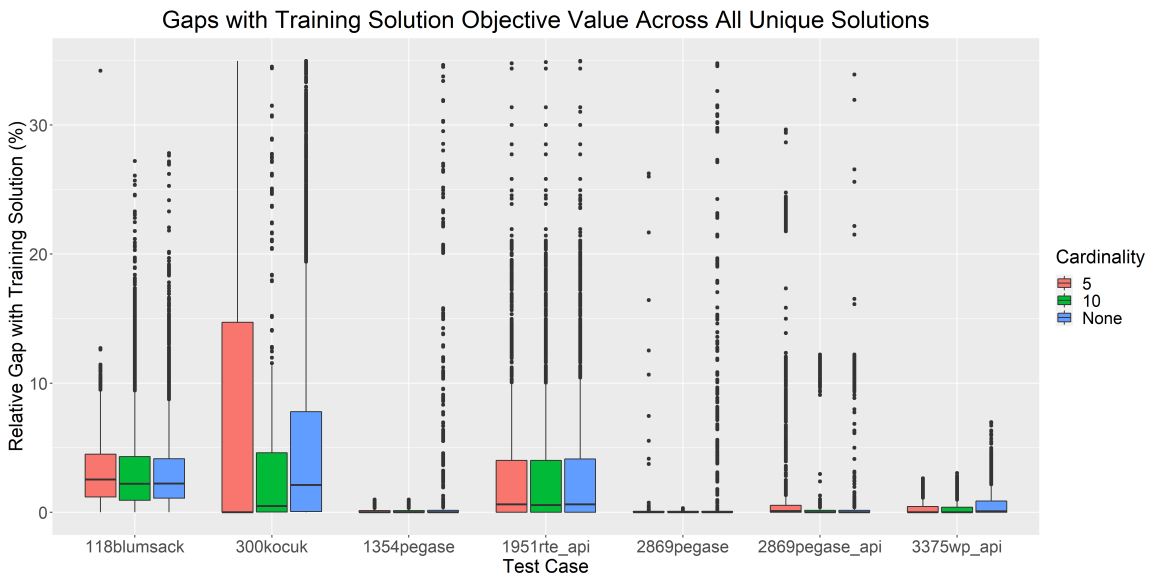


Figure 2.9: The distribution of the quality of all unique feasible training solutions for all 30 test instances. Note that many outliers (often corresponding to solutions with high load shed or over-generation) are omitted.

ery unique topology from the training set in each of the test instances (i.e., if there are 50 unique solutions, since there are 30 test instances, this is a distribution of 1,500 gaps). We can see that, while there is a lot of variation among the test cases, most of the training solutions are at least 5%-optimal for the test instances. However, there are many outliers which are far worse, usually because they are actually infeasible (i.e., have positive load shed or over-generation). On average, these gaps are higher than the gaps of less than 0.75% which we achieve from the KNN heuristic, so despite the sparsity of the set of lines which are opened, the distance between instances in parameter space still plays a role in finding high-quality primal solutions.

2.5.2 Is Distance the Correct Metric?

To explore the role of distance in determining instances with the same DCOTS solution, we again considered the set of all unique solutions for a fixed test case and cardinality variation. This time, using the Euclidean norm, we calculated the cardinal distance of the best training solution from among all the training solutions. (By cardinal distance, we

mean the integer number where 1 is the closest of the instances, 2 is the second closest, and the furthest is the number of unique solutions.) The distributions of the cardinal distance of the best solution is shown in Figure 2.10. Surprisingly, the best solutions do not tend to be among the closest, with the exception of the 300kocuk case with cardinality 5. For most of the instances, the best solution is fairly evenly distributed (keeping in mind that the maximum value for an individual box plot in the figure is the number of unique solutions for that instance). This figure suggests that KNN is not the right heuristic for finding the best solution in the training set: There is no reasonable value of k which could consistently find the best solution.

However, Figure 2.10 obscures the fact that, just because the best solution is far does not mean that near solutions are not good. In Figure 2.11, we plot the cardinal distance of the closest solution in the training set which is at least 1%-optimal. The horizontal line is drawn at the cardinal distance of 10. This figure illuminates why the KNN heuristic is successful despite what we show in Figure 2.10: Within the neighborhood defined by setting $k = 10$, with few exceptions, we have a 1%-optimal solution. This stems partly from what we observed in Figure 2.9: In four of our test systems, more than half of the solutions in the training set are within 1% of optimality for the instances in the test set. Note, however, that this is not true for 118blumsack, 300kocuk, or 1951rte_api, and for these test systems also, $k = 10$ is sufficient to find high-quality solutions.

2.5.3 Buses with Similar Effect on Congestion

While we have seen that distance does work as a metric to find high-quality solutions within the training set, we also suspect that insight into which changes in demand lead to congestion in the network and where this congestion is would be more powerful in determining a DCOTS solution from the training set. Essentially, understanding what directions in the parameter space exacerbate or reduce certain congestion patterns might mean that the distance between a test instance and a training instance is not as important in determining

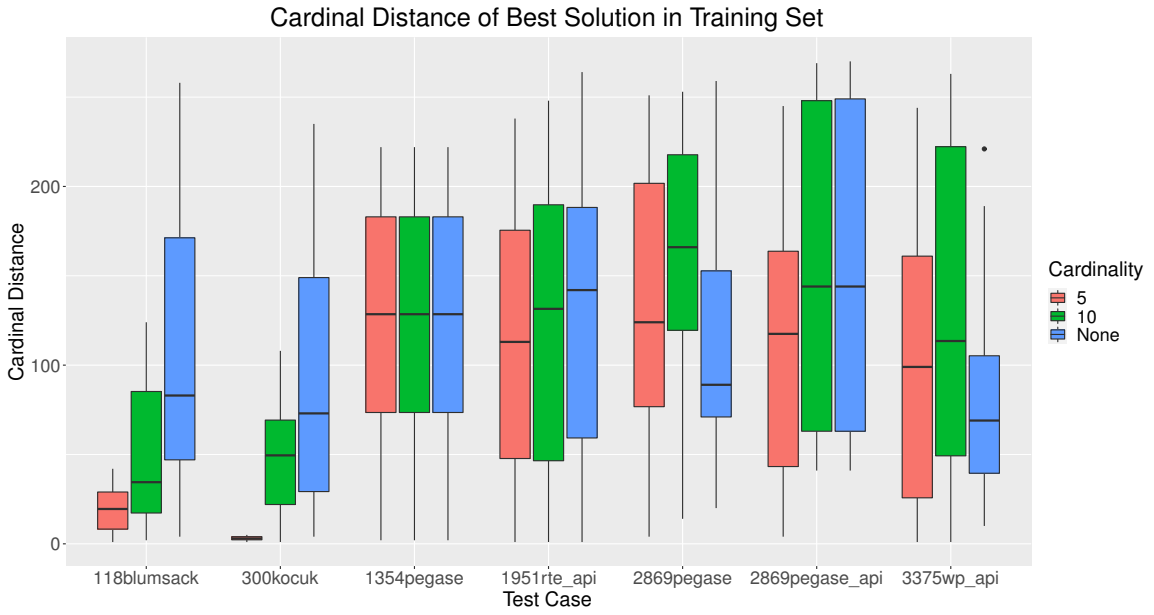


Figure 2.10: The distribution of the cardinal distances of the best solution in the training set for the 30 test instances in each test case and cardinality variant. The best solution does not tend to be close to the test instance.

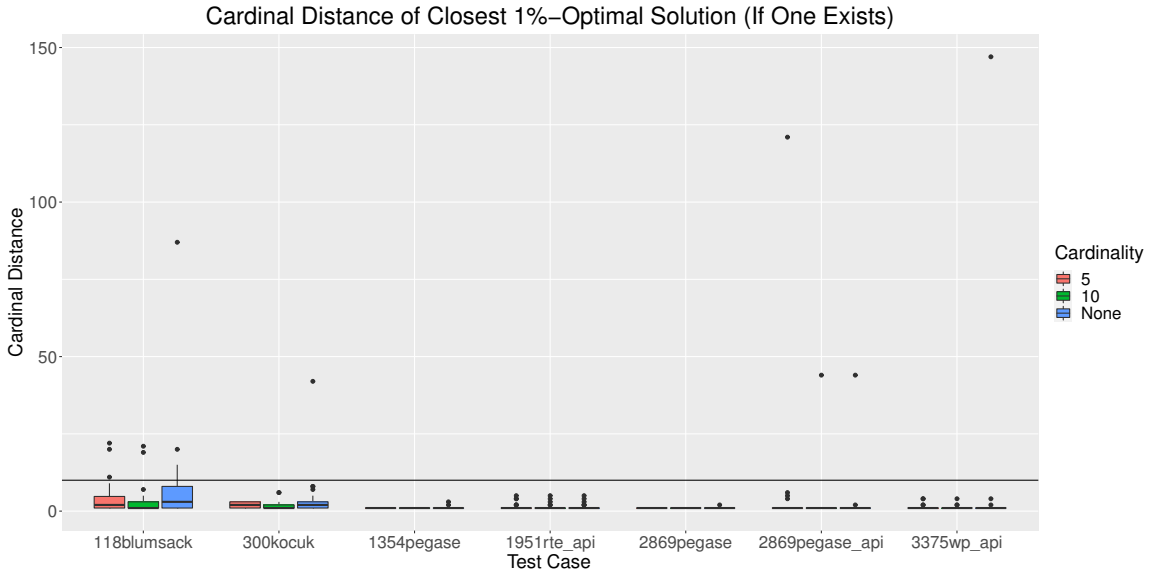


Figure 2.11: The distribution of the closest 1%-optimal solution in the training set for the 30 test instances in each test case and cardinality variant. The black horizontal line denotes a cardinal distance of 10.

the quality of the training DCOTS solution as the direction of the vector between the two instances is. If the way the demands have shifted does not change where the network is congested, the same topology may be near-optimal. While we do not have a precise characterization of these directions, in this section we show that changes in demand in some parts of the network have identical effects on the DCOTS solution as if those changes had been in other parts of the network.

We present results from an experiment where, for each bus in the 118blumsack network, we increased its demand by 1 p.u. above the nominal demands and found a 1%-optimal transmission switching solution with cardinality 5. We then sorted the buses of the network into classes that shared the same 1%-optimal transmission switching solution in the above experiment. The results are visualized in Figure 2.12. The largest three classes are colored, and buses indicated by dotted lines have a corresponding switching solution which differs by only one line from the class sharing their color. The lines which are switched in any of these solutions are also labeled and colored according to one of the solutions in which they are opened.

When we sort the buses in this way, they divide regionally. In this network, the lower right-hand region's demand far exceeds its generation capacity, meaning that power must move downward through buses 77 and 80. This creates congestion in the part of the network which divides the blue and green classes from the red one. This diagram leads us to hypothesize that the optimal DCOTS solution is not sensitive to which buses have increased demand within a class, only to whether there is an increase regionally. In other words, the buses within a class behave like each other in terms of their effect on the overall congestion pattern and hence on the DCOTS solution.

To test this hypothesis, we reran the KNN heuristic on data where only the demand varied and the generation costs were constant. Instead of characterizing an instance by its vector of demands, we aggregated the net demand within each class (including 2- and 3-bus classes not pictured in Figure 2.12). For the 118blumsack instance, this means that

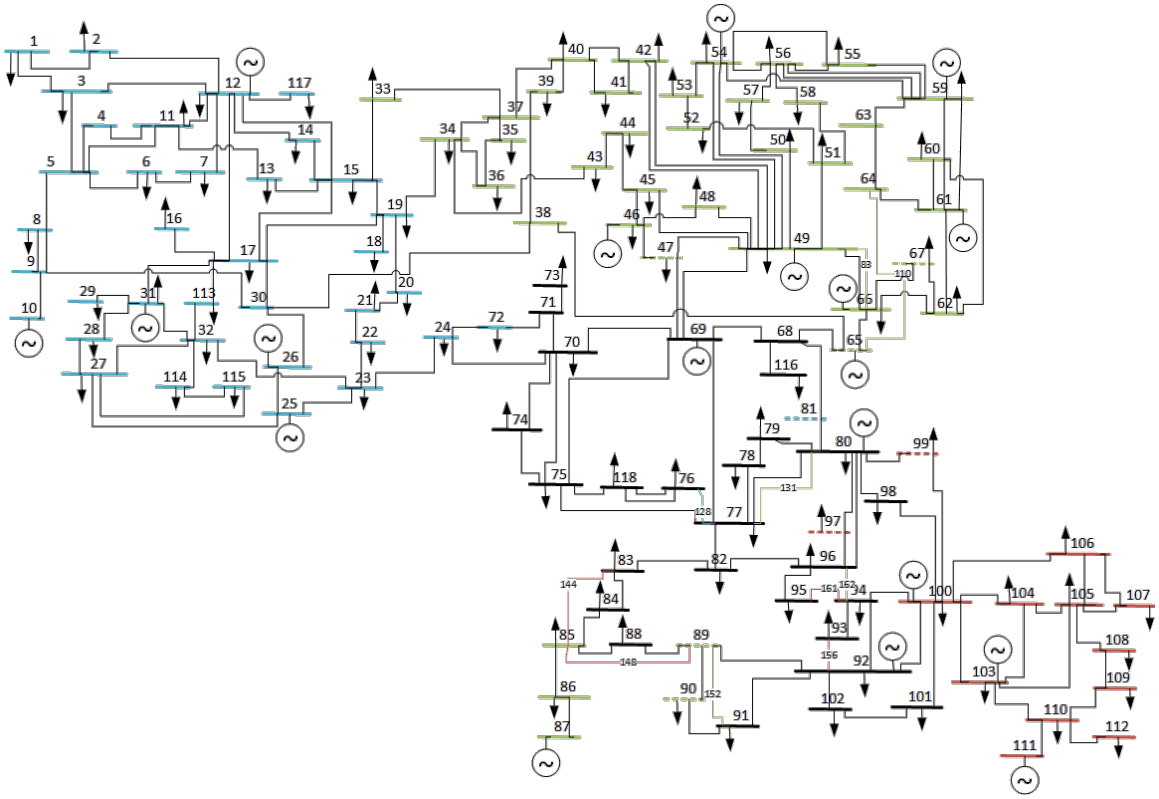


Figure 2.12: Buses of the 118blumsack network labeled by their optimal switching solution when their demand is increased by 1 p.u. The 35 buses colored green (in the upper-right region) all switch lines 83, 110, 131, 152, and 162 (labeled and also colored green) in their corresponding optimal solution. The 37 buses labeled blue switch lines 110, 128, 131, 152, and 162 in their corresponding optimal solution. The 11 buses colored red switch lines 144, 148, 156, 161, and 162. The buses represented with dotted lines correspond to optimal switching solutions which differ from their color by only one line. Uncolored buses belong to either 1-, 2-, or 3-bus classes not shown in this diagram.

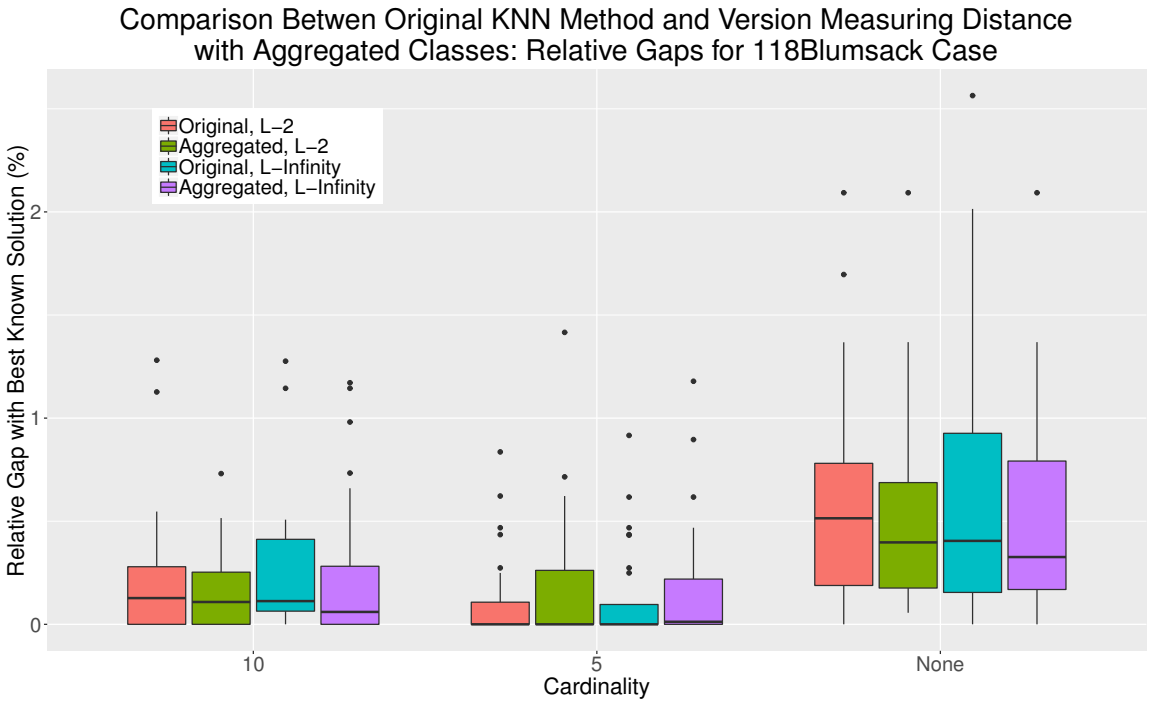


Figure 2.13: Results comparing solution quality of the KNN heuristic using the original method and using a reduced-dimension vector of net demands in each of the classes from Figure 2.12.

instead of a parameter vector of length 118, we used one of length 30. The results are compared with the original method (still with only variance in demand, not generation costs) in Figure 2.13. For this instance, the reduced-dimension version of KNN performs on average as well as the original version, and, for the larger cardinalities, has slightly reduced variance. In experiments not reported here, we tried the same procedure on the 300kocuk case, with similar results.

We do not show results for this experiment for larger test systems because the training to find the classes is much more computationally intensive, as it scales with the number of buses in the network. We have no reason to believe that this method is an improvement over the original method, given the difficulty of training and the fact that the computational time of the heuristic is not significantly reduced. (The distance calculations do not dominate the time: The LP solves do.) However, we present these results because they are revealing of what information matters in determining an optimal topology: Many buses behave like

each other when it comes to how their demands affect the DCOTS solution. In particular, it appears that changes in less-congested regions of the network can be treated in aggregate. What is important in determining the optimal DCOTS solution is changes in the overall patterns of congestion.

2.6 Alternatives to KNN

The analysis from the previous section raises further questions about variations on the KNN heuristic:

1. Could alternative feature spaces improve the KNN heuristic? And further, could these be based on the dual information of the DCOPF problem, like the heuristics from Ruiz, Foster, et al. (2012)?

2. In light of the results shown in Figure 2.9, for values of k as large as 10, is the KNN heuristic better than choosing the best of k random solutions from the training set?

In this section, we examine these questions. We find that several alternative feature spaces perform as well as the KNN heuristic proposed in this chapter, but that none are consistently better. We also show that, indeed, choosing the best solution from k random training solutions performs comparably to the KNN heuristic for larger values of k. We show also that the quality of solutions from this method degrades as k is reduced, which we would also expect based on Figure 2.9.

2.6.1 Alternative Feature Spaces for the KNN Heuristic

In this section, we present results where, instead of measuring the distance between instances of DCOTS in the space of the demands and generations costs, we will do so in features spaces defined by various dual information from solving the DCOPF problem on the network, fixing the demands and generation costs that we were previously using to define the instance. In particular, we try defining the feature space in three ways. Each of

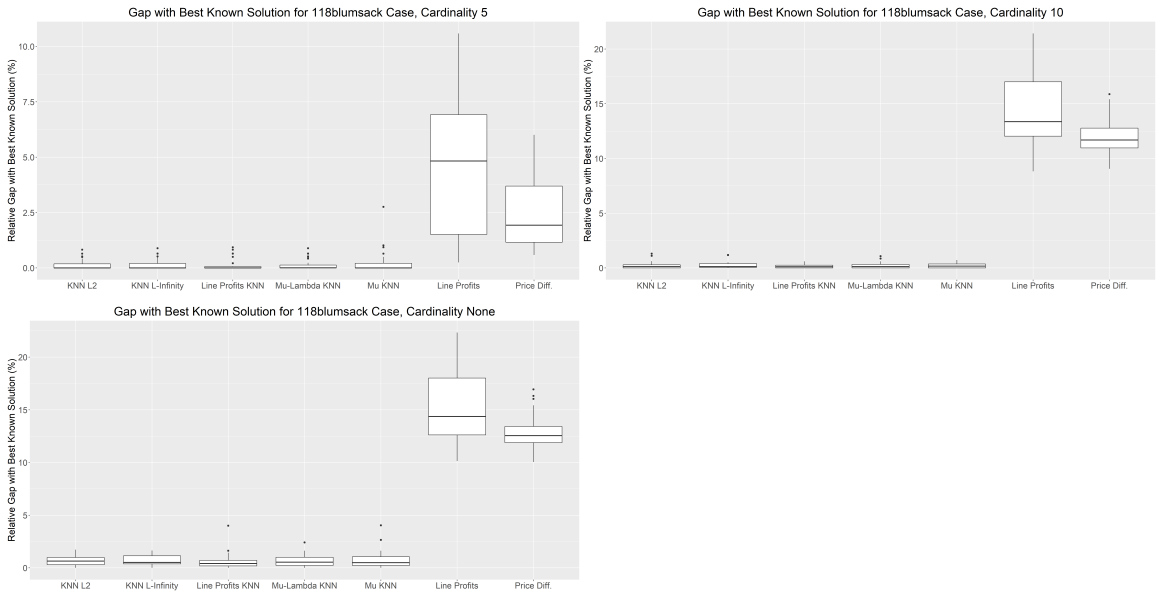


Figure 2.14: Solution quality of the three KNN heuristic variations as compared to the original KNN heuristics and two of the better-performing heuristics from Ruiz, Foster, et al. (2012) on the 118blumsack case.

these involves solving DCOPF with the instance's demands and generation costs and using the resulting primal and dual solutions. In the first, for each instance we use a vector of length $|\mathcal{L}|$ where each entry is the Line Profits criterion from Ruiz, Foster, et al. (2012) for the corresponding line. That is, the entry corresponding to line l is calculated as

$$f_l(\mu_{d(l)} - \mu_{o(l)}),$$

where f_l is the flow on line l from the primal DCOPF solution and $\mu_{o(l)}$ and $\mu_{d(l)}$ are the duals on the nodal balance constraints corresponding to the origin and destination nodes of line l. In the second, we append the duals of the thermal limit constraints, λ^+ and λ^-, and the nodal prices, μ, and normalize the resulting vector. In the last, we use only the vector μ.

We show results for the three experiments above compared to the KNN heuristics as well as the Line Profits and Price Difference versions of the heuristics from Ruiz, Foster, et al. (2012). As before, we plot the distribution of relative gaps of the best known solution for the 30 testing instances.

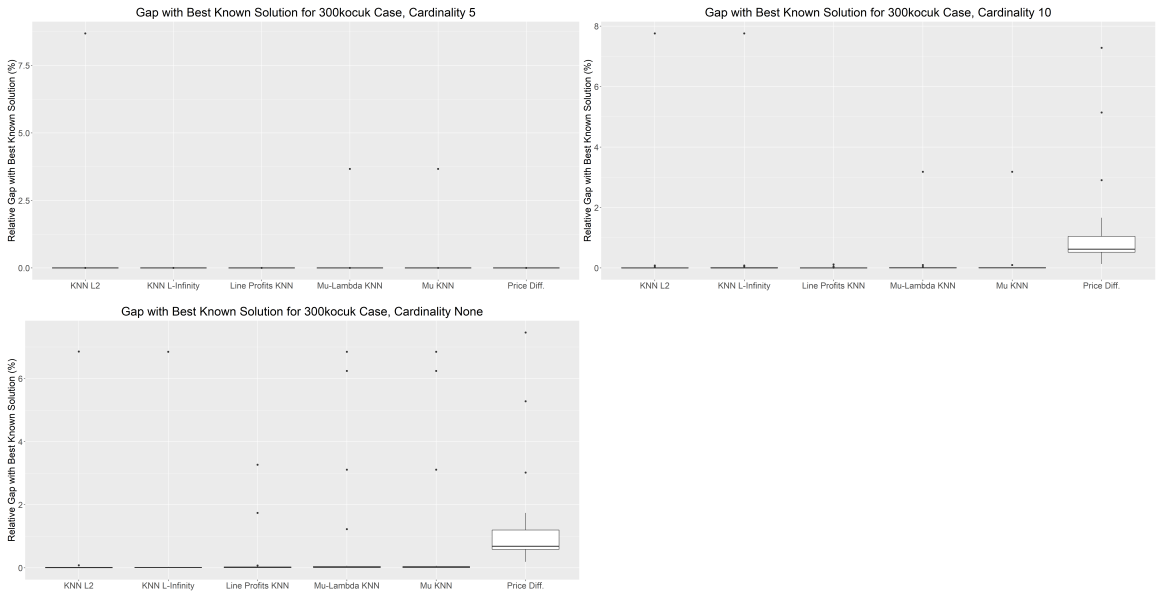


Figure 2.15: Solution quality of the three KNN heuristic variations as compared to the original KNN heuristics and the Price Difference heuristic from Ruiz, Foster, et al. (2012) on the 300kocuk case. We leave out the Line Profits heuristic in this plot since it is not competitive for the 300kocuk case (see Figure 2.2).

The results for each cardinality of each of the test cases are shown in Figures 2.14-2.20. The three alternative features spaces are labeled as "Line Profits KNN," "Mu-Lambda KNN," and "Mu KNN" respectively. We see that for most of the test cases, the results for all three are roughly the same as each other and as the original KNN heuristics. In the 2869pegase_api case only (Figure 2.19), all three of the variants consistently outperform the original versions for both cardinality constraints of 5 and 10 lines.

Overall, none of the three variations on using the dual information of DCOPF to define the KNN feature space seems to outperform any of the others, and even in the case of the 2869pegase_api case, the superiority in performance is not statistically significant. Thus, it seems reasonable to conclude that alternative notions of the distance between instances of the DCOTS problem could be effective, but it is not clear which is best. These results are somewhat surprising given that Proposition 1 gives some theoretical justification as to why the original KNN heuristics could work. However, the impressive quality of the solutions from particularly the Line Profits criterion version of the heuristics from Ruiz, Foster, et al.

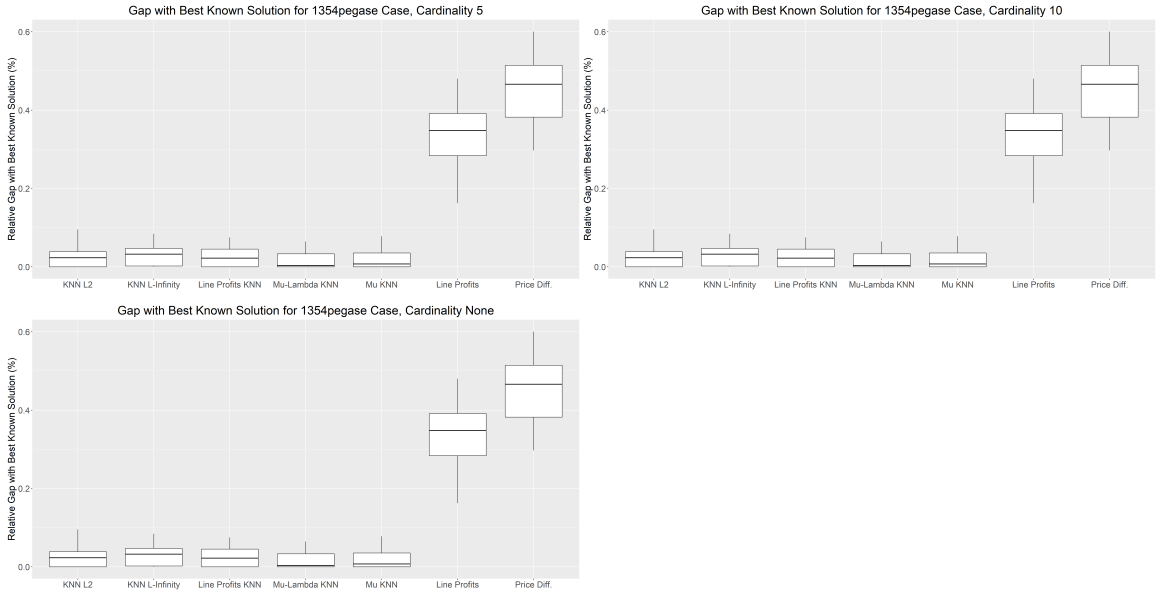


Figure 2.16: Solution quality of the three KNN heuristic variations as compared to the original KNN heuristics and two of the better-performing heuristics from Ruiz, Foster, et al. (2012) on the 1354pegase case.

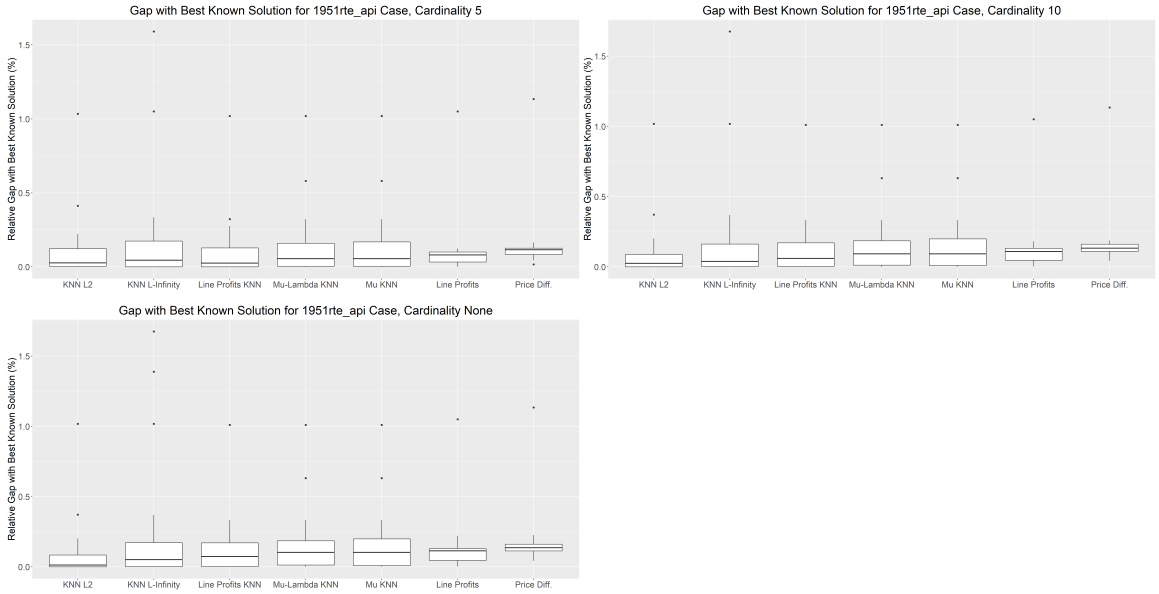


Figure 2.17: Solution quality of the three KNN heuristic variations as compared to the original KNN heuristics and two of the better-performing heuristics from Ruiz, Foster, et al. (2012) on the 1951rte_api case.

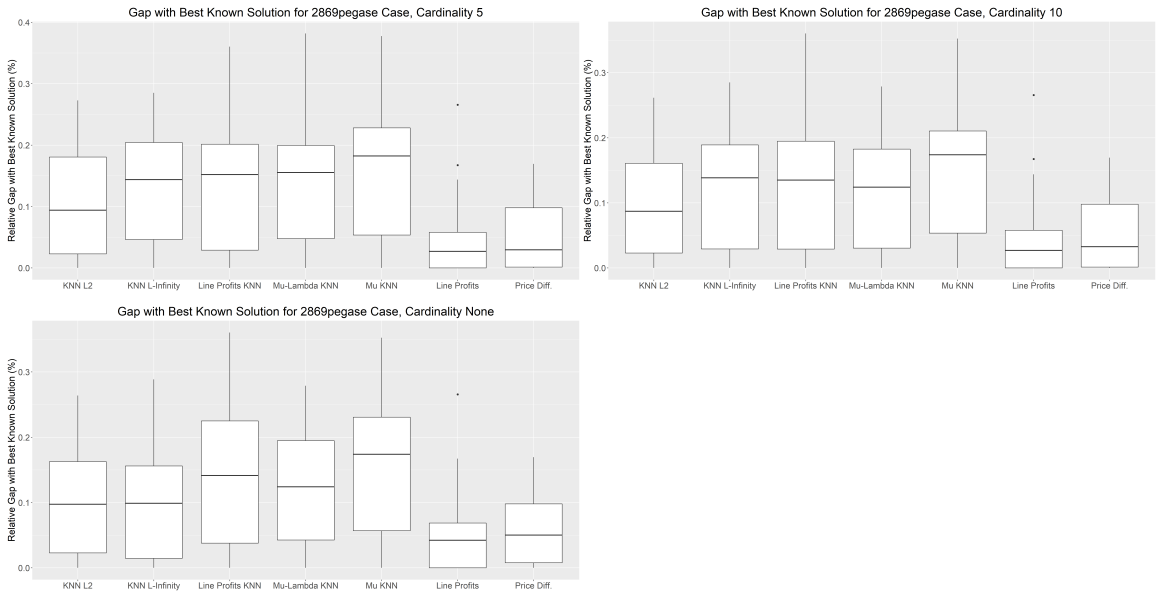


Figure 2.18: Solution quality of the three KNN heuristic variations as compared to the original KNN heuristics and two of the better-performing heuristics from Ruiz, Foster, et al. (2012) on the 2869pegase case.

(2012) do show the criterion to be a useful indicator of where the network is congested, and it seems that the dual information it relies on is also able to convey much of the same information.

2.6.2 K Random Neighbors

In this section, we address our second question and compare to an algorithm we will refer to as k *random neighbors*. In this approach, we first select k random instances from the training set as the set of instances T and then follow the procedure from lines 12-18 of Algorithm 4. We show results first using $k = 10$, as in the results we presented in Section 2.4, and then with $k = 3$.

Figure 2.21 shows a comparison on the 118blumsack case between the optimality gaps from the k random neighbors heuristic, the KNN heuristics, and the Line Profits and Price Difference criteria versions of the heuristics from Ruiz, Foster, et al. (2012) for both $k = 10$ and $k = 3$. Note that, because we are plotting a relative gap with the best known solution, the distributions for the heuristics from Ruiz, Foster, et al. (2012) change between the

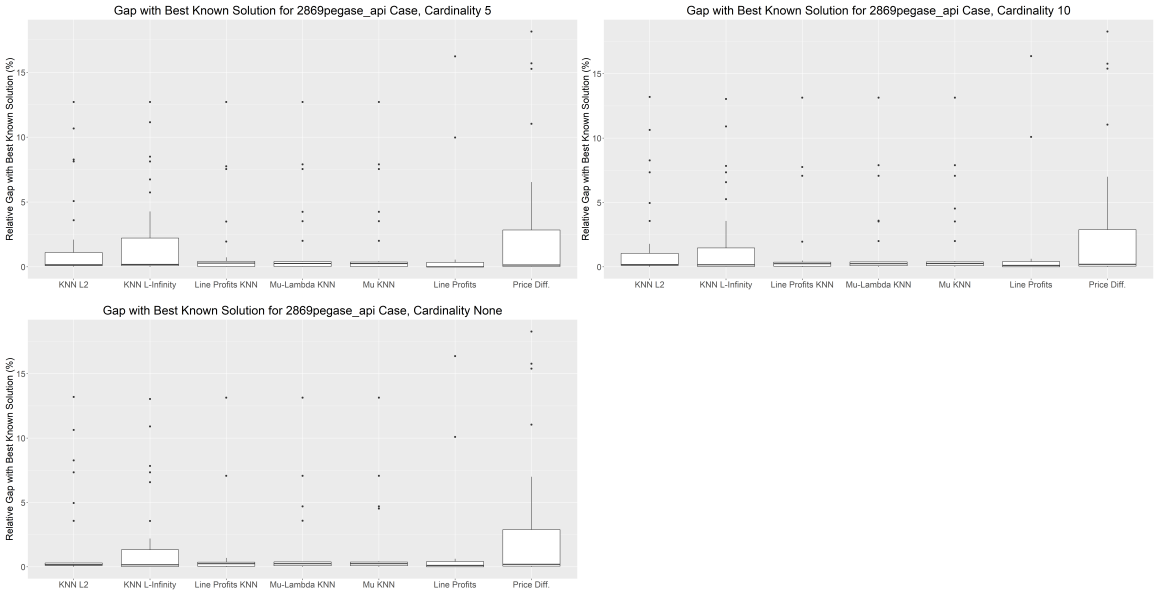


Figure 2.19: Solution quality of the three KNN heuristic variations as compared to the original KNN heuristics and two of the better-performing heuristics from Ruiz, Foster, et al. (2012) on the 2869pegase_api case.

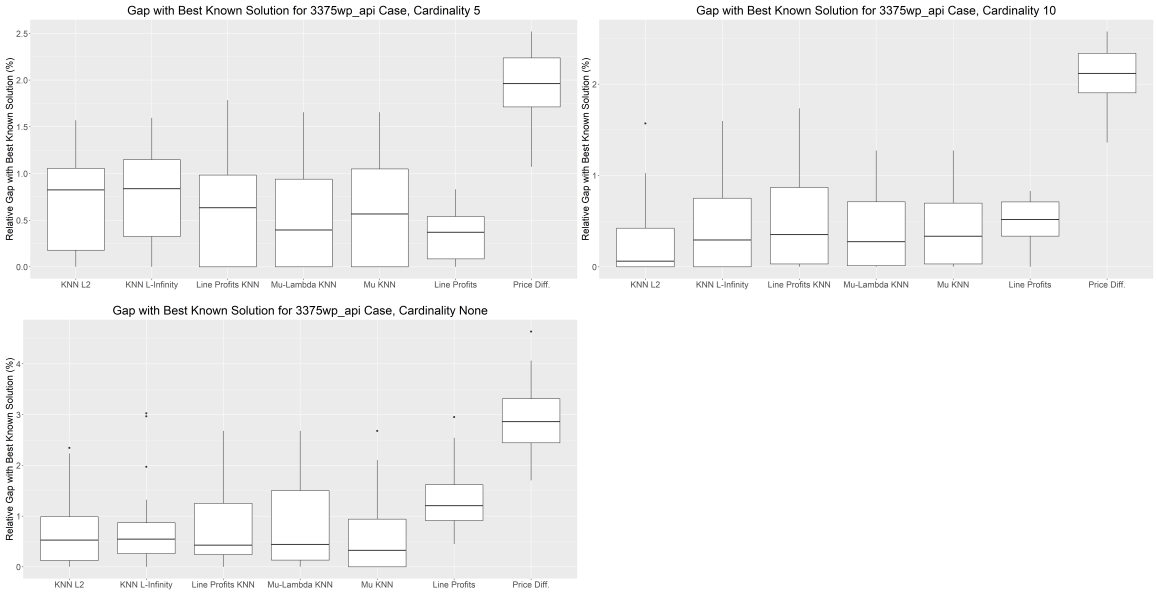


Figure 2.20: Solution quality of the three KNN heuristic variations as compared to the original KNN heuristics and two of the better-performing heuristics from Ruiz, Foster, et al. (2012) on the 3375wp_api case.

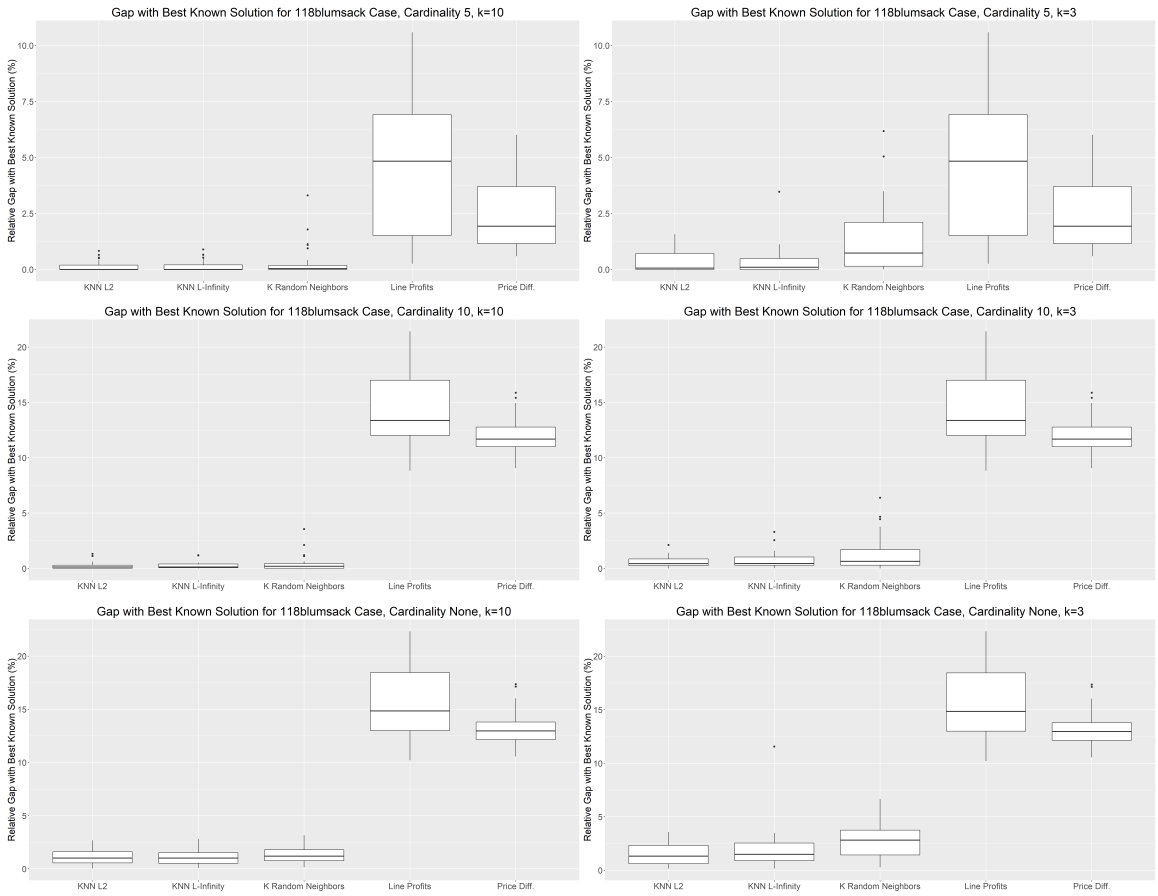


Figure 2.21: Solution quality of the k random neighbors heuristic as compared to the KNN heuristics and two of the better-performing heuristics from Ruiz, Foster, et al. (2012) on the 118blumsack case. The first column of figures uses $k = 10$ for the first three heuristics, and the second column uses a value of $k = 3$.

columns also, since the solutions generated from the KNN heuristics and the k random neighbors heuristic are different. We see that for $k = 10$, there is very little difference in k random neighbors and the KNN heuristics, though k random neighbors can result in some worse outliers. However, for $k = 3$, there is noticeable deterioration in the quality of the solution from the k random neighbors heuristic. This aligns well with what we saw in Figure 2.9 since, for the 118blumsack case in that figure, the majority of the solutions in the training set are within 5% of the best known solution, so sampling 10 randomly is enough to find a relatively good one. However, sampling only 3 randomly is not quite as successful. Note however, that even with $k = 3$ the k random neighbors heuristic performs better than either of the Line Profits or Price Difference heuristics.

We plot the same comparison for the other test cases in Figures 2.22-2.27. In the 300kocuk case (Figure 2.22), the k random neighbors heuristic is again comparable to the previous heuristics when $k = 10$, but for $k = 3$, it is slightly worse. This is expected given that, in Figure 2.9, a larger portion of the distribution of training solution quality is worse than 5% optimal. In the 1354pegase case, the difference between $k = 10$ and $k = 3$ for the k random neighbors heuristic is very slight: In both cases it performs comparably to the KNN heuristics, and much better than the Line Profits and Price Difference heuristics. This again makes sense in light of Figure 2.9 since, for this test case, nearly all of the training solutions are within 1% of the quality of the best known solution, so even sampling 3 randomly is expected to perform well. In the 1951rte_api case, the k random neighbors heuristic actually outperforms both the KNN heuristics with $k = 10$, making it the only heuristic significantly better than the Line Profits and Price Difference heuristics. Even with $k = 3$, it still noticeably outperforms the ℓ_∞-norm variant of the KNN heuristic, though its performance is this time comparable to the heuristics from Ruiz, Foster, et al. (2012). Although it is not clear why, it seems that distance may somehow be a misleading metric for this particular test network. In addition, it is unsurprising that there is a difference between the performance for $k = 10$ and $k = 3$ since a greater proportion of the

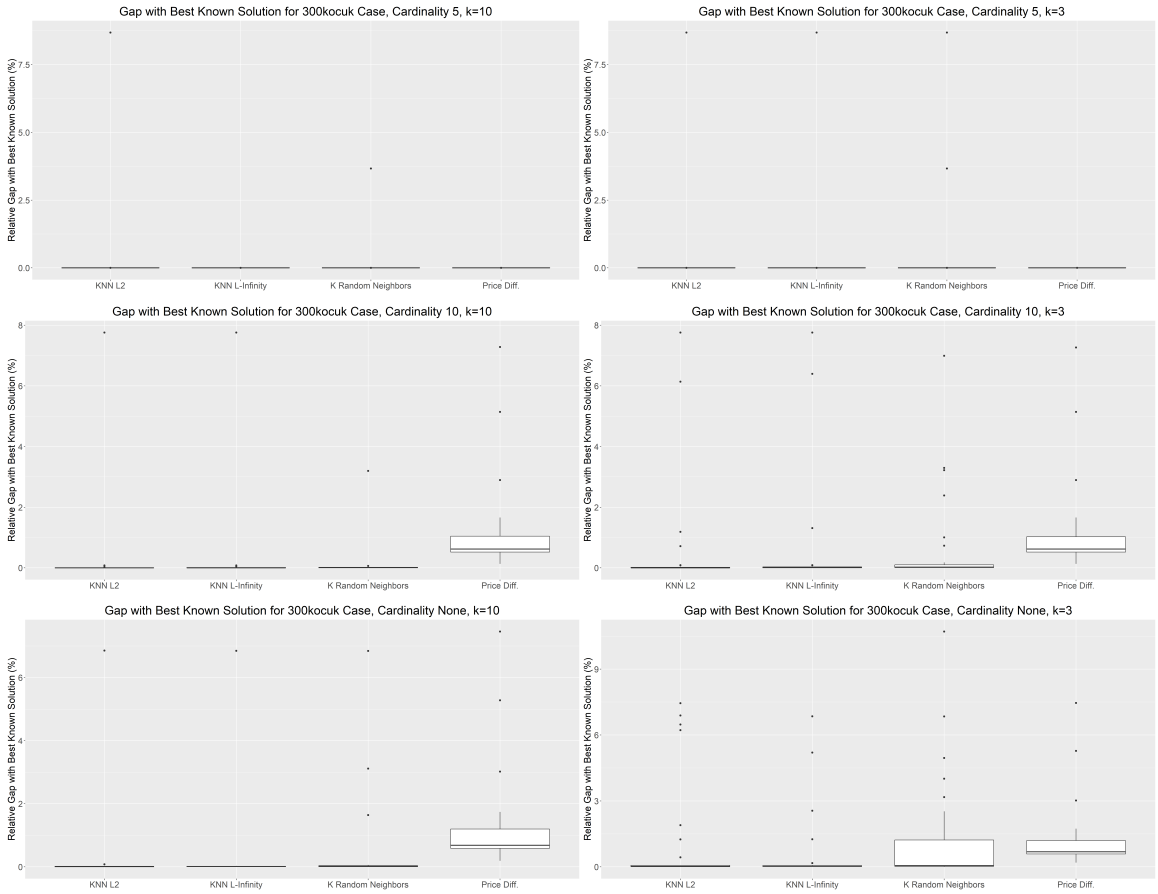


Figure 2.22: Solution quality of the k random neighbors heuristic as compared to the KNN heuristics and the Price Difference heuristic from Ruiz, Foster, et al. (2012) on the 300kocuk case. The first column of figures uses $k = 10$ for the first three heuristics, and the second column uses a value of $k = 3$. We again leave out the Line Profits heuristic in these plots, as it performs poorly for this test case.

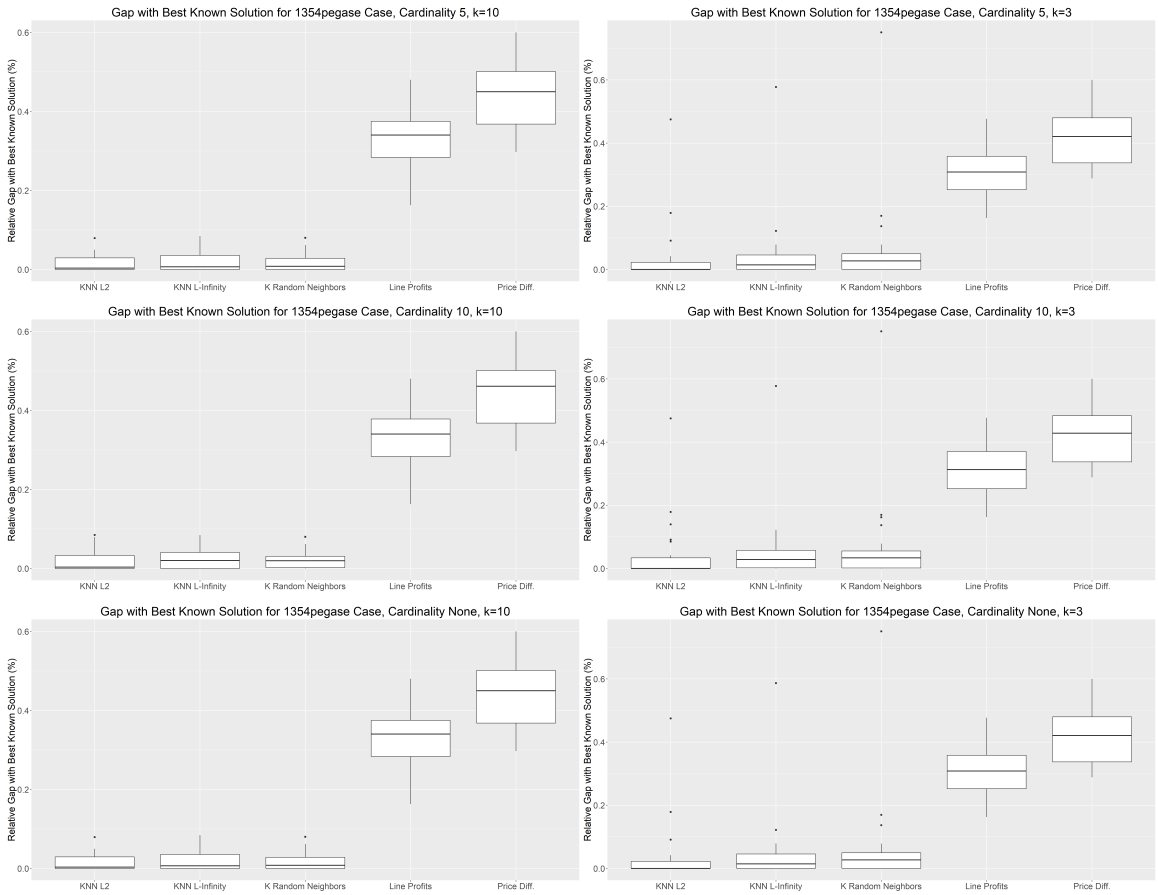


Figure 2.23: Solution quality of the k random neighbors heuristic as compared to the KNN heuristics and two of the better-performing heuristics from Ruiz, Foster, et al. (2012) on the 1354pegase case. The first column of figures uses $k = 10$ for the first three heuristics, and the second column uses a value of $k = 3$.

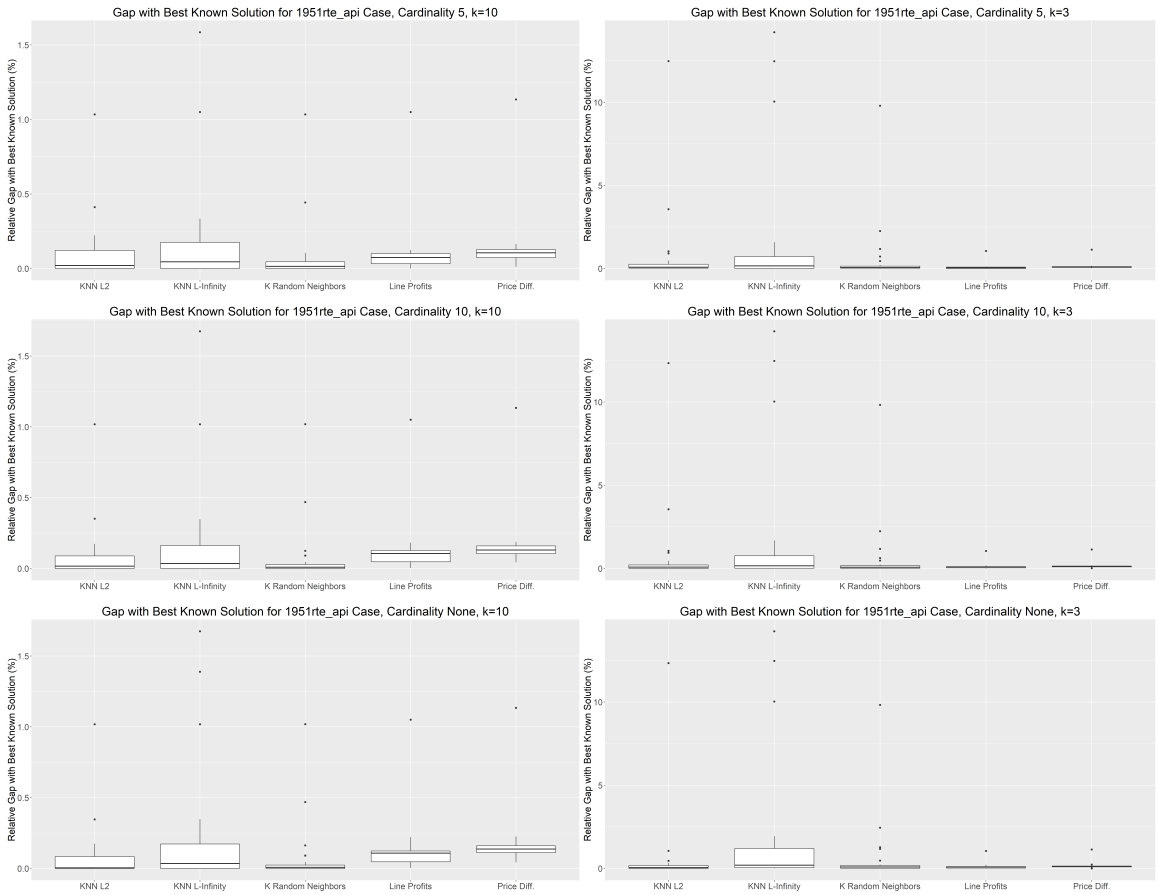


Figure 2.24: Solution quality of the k random neighbors heuristic as compared to the KNN heuristics and two of the better-performing heuristics from Ruiz, Foster, et al. (2012) on the 1951rte_api case. The first column of figures uses $k = 10$ for the first three heuristics, and the second column uses a value of $k = 3$.

solutions for the 1951rte_api case are between 1% and 5% of the best known solution in Figure 2.9. In the 2869pegase case and in the 2869pegase_api case, shown in Figures 2.25 and 2.26 respectively, k random neighbors is slightly worse, though not significantly so, than the KNN heuristics. The quality of all three declines for $k = 3$, and is worse than the Line Profits and Price Difference heuristics. The results for the 3375wp_api case are again similar: In Figure 2.27 the quality of the solutions from k random neighbors is slightly worse than the KNN heuristics. They are mainly comparable to the Line Profits heuristic when $k = 10$, but not when $k = 3$

In conclusion, we see that for most of the test cases, the k random neighbors heuristic with $k = 10$ is comparable with the performance of the KNN heuristics for the same value

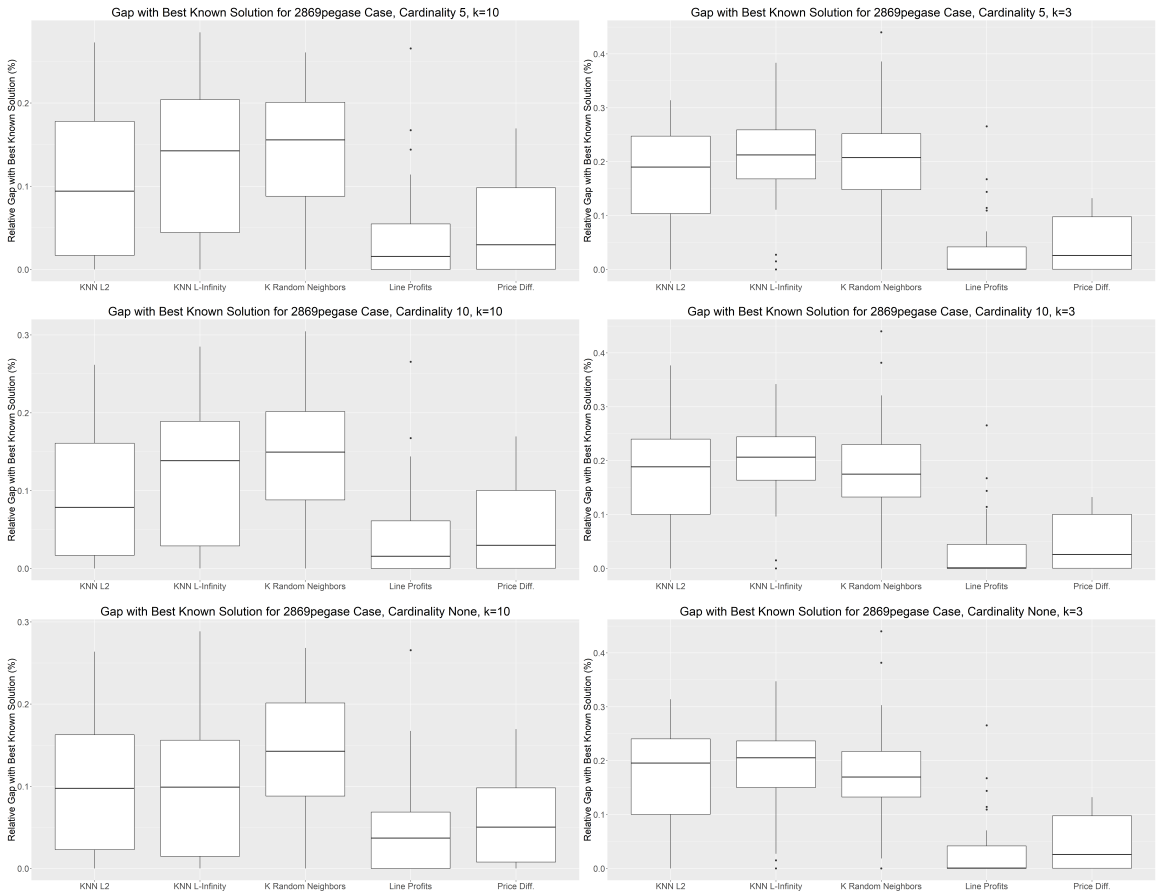


Figure 2.25: Solution quality of the k random neighbors heuristic as compared to the KNN heuristics and two of the better-performing heuristics from Ruiz, Foster, et al. (2012) on the 2869pegase case. The first column of figures uses $k = 10$ for the first three heuristics, and the second column uses a value of $k = 3$.

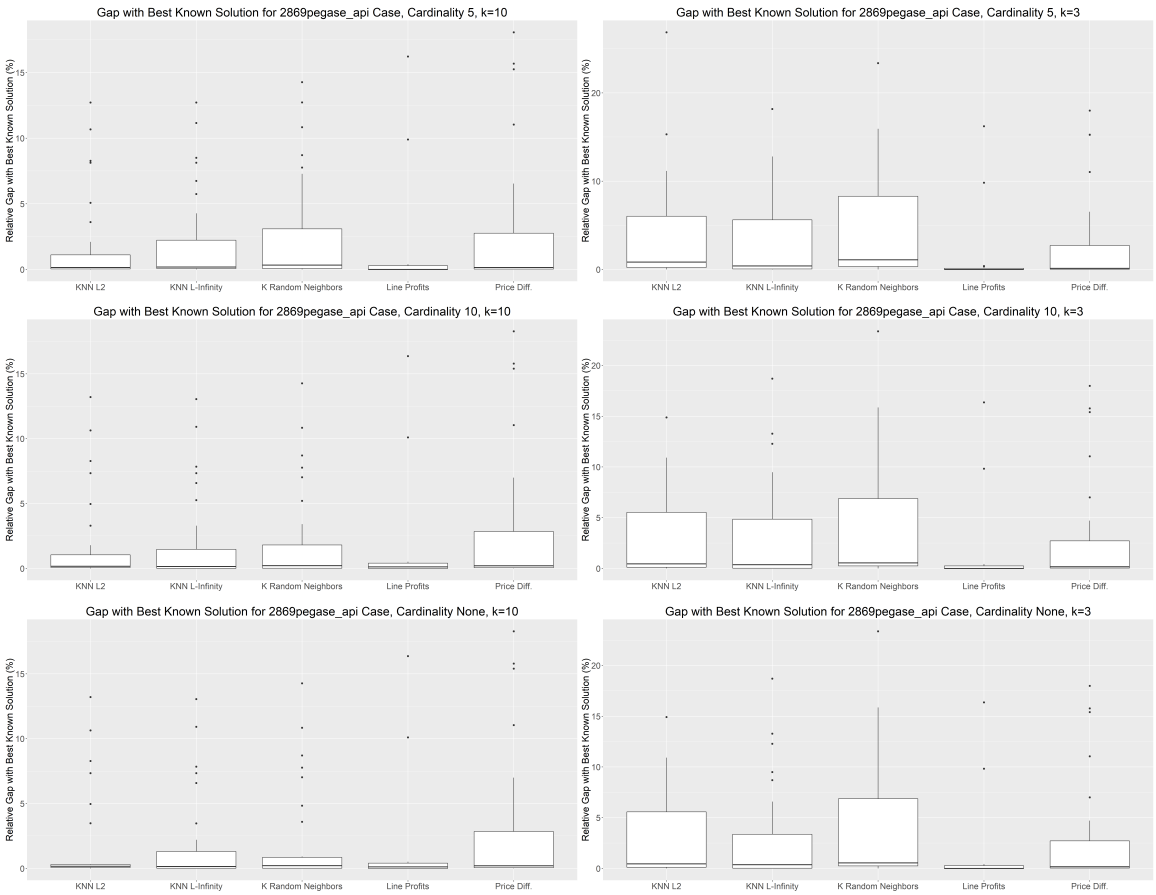


Figure 2.26: Solution quality of the k random neighbors heuristic as compared to the KNN heuristics and two of the better-performing heuristics from Ruiz, Foster, et al. (2012) on the 2869pegase_api case. The first column of figures uses $k = 10$ for the first three heuristics, and the second column uses a value of $k = 3$.

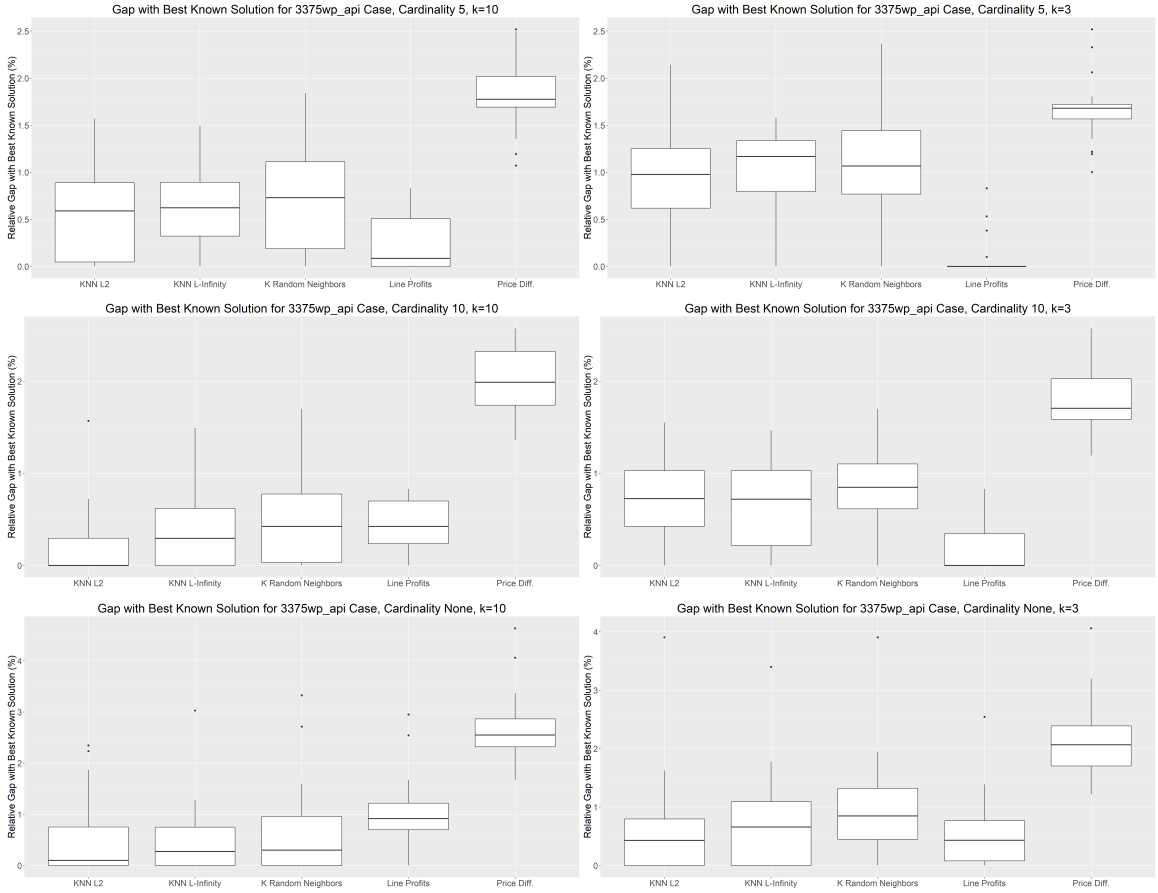


Figure 2.27: Solution quality of the k random neighbors heuristic as compared to the KNN heuristics and two of the better-performing heuristics from Ruiz, Foster, et al. (2012) on the 3375wp_api case. The first column of figures uses $k = 10$ for the first three heuristics, and the second column uses a value of $k = 3$.

Table 2.5: Number of test instances for which each heuristic finds the best solution out of all the heuristic solutions. Ties are counted as both of the heuristics finding the best solution, explaining why the sums of the rows can exceed 30.

Test Case	Cardinality	KNN L2	KNN L-Infinity	Line Profits KNN	Mu-Lambda KNN	Mu KNN	KRN	Line Profits	Price Diff.
118blumsack	5	9	9	17	14	14	13	0	0
	10	6	5	14	10	11	6	0	0
	None	8	7	13	7	10	4	0	0
300kocuk	5	16	16	13	13	13	13	1	4
	10	13	10	12	4	6	5	0	0
	None	11	12	5	4	4	6	0	0
1354pegase	5	7	8	7	11	11	6	0	0
	10	7	8	7	11	11	6	0	0
	None	7	8	7	11	11	6	0	0
1951rte_api	5	5	8	11	10	10	8	1	0
	10	11	6	6	5	5	7	0	0
	None	10	7	6	5	6	8	0	0
2869pegase	5	6	3	2	2	1	1	11	9
	10	5	3	2	3	1	1	11	9
	None	5	6	1	2	0	2	10	7
2869pegase_api	5	0	3	7	8	8	3	16	0
	10	3	5	3	5	4	6	12	0
	None	3	5	3	5	4	6	12	0
3375wp_api	5	5	2	9	12	10	6	5	0
	10	13	9	7	8	7	5	1	0
	None	5	7	7	7	9	7	0	0
Total Over All Test Cases	5	48	49	66	70	67	50	34	13
	10	58	46	51	46	45	36	24	9
	None	49	52	42	41	44	39	22	7
Total		155	147	159	157	156	125	80	29

of k. With the exception of the 1951rte_api instance, this heuristic is not an improvement, but the fact that it performs well suggests again that our notion of distance in KNN could be improved.

2.6.3 Is KNN the Correct Approach?

In Table 2.5, we compare the three variations of KNN from Section 2.6.1 and the k-random neighbors heuristic in the same format as we used in Table 2.2. We also include the Line Profits and Price Difference heuristics from Ruiz, Foster, et al., 2012 for comparison. Overall, the Line Profits criterion version of KNN finds the best-known solution most often, but all of the KNN variants are in close competition. The dual-information-based variants of

KNN tend to perform much better than all the others when only 5 lines can be switched, but they are not necessarily better for the less-restrictive cardinality constraints. Last, k-random neighbors performs better than the heuristics from Ruiz, Foster, et al., 2012, but not as well as any of the KNN variants. This shows that our notions of distance are helping to guide the KNN heuristics to good solutions, but also that the structure of our problem and our data are helping the performance of the KNN heuristics as well.

Overall, in this section we see results that echo those of Section 2.5. Specifically, first, while the notion of distance in our proposed KNN heuristic is both functional and mathematically supported by Proposition 1, using the KNN heuristic with alternative feature spaces can lead to equally high-quality solutions. Second, as Figure 2.9 suggests, on our test networks using $k = 10$, it is possible to find good solutions without a notion of distance: The likelihood of finding 10 poor-quality solutions in the training set is so low that randomly sampling is effective. This aligns with the conclusion from Figure 2.10 that the notion of distance in the KNN heuristic does not lead us to the best solutions in the training set. It also suggests what we observed in Section 2.5.1 that what lines are good to open seems to be more a property of the physical network itself, and hence does not seem sensitive to changes in demand and generation cost. However, as we see in Table 2.5, the KNN variants are slightly better-performing than the k-random neighbors heuristic. Thus, this section suggests both that different feature spaces could be used successfully in the KNN heuristic, but also that there is a need for a deeper understanding of how the physical grid and nominal demands and generation costs impact what makes a good topology. In summary, these results still suggest that the KNN heuristic is a reasonable approach, but show that there are other ways to take advantage of the relatively small variance in demands and generation costs and the effects of the grid structure on choosing good topologies.

2.7 Conclusion

We presented a KNN-based heuristic for DCOTS which finds high-quality solutions within the time limit imposed by real-time. We showed through a case study on seven test instances that this heuristic yields solutions competitive with heuristics from the literature, and in less computational time. In particular, the heuristic scales up well with the size of the network since it has only a weak dependence on the number of lines in the system. Last, through an analysis of our training data, we observe that relatively few lines are ever opened in any DCOTS solutions for a fixed network. Despite this, there is variation in the quality of the optimal topologies in the training solutions for the test instances. While distance is an imperfect metric for finding the best topology, it is proficient at finding high-quality topologies. However, it would be of interest to be able to characterize the effects on congestion of both the original structure of the grid and changing demands, as these relationships appear to drive the optimal DCOTS solution.

Since transmission switching is a tool to reduce generation costs given real-time fluctuation in demand, it is a problem which in practice should be solved quickly, but one for which data from past solves is plentiful. This makes it an ideal problem for machine learning techniques. In the future, this same heuristic could be applied to AC optimal transmission switching, and potentially to security-constrained DCOTS. In addition, it could be used to find good warmstarts for planning problems or day-ahead operational problems in power systems. It remains an open question how increased variance in the demands and generation costs (which are not unreasonable in a future with higher renewable energy penetration) would impact the performance of the KNN heuristic. Similarly, the interaction between storage systems and transmission switching will likely be of interest, and the KNN heuristic could be used in an integrated model of the two. Last, for many of these problems, extensions to multi-period problems will be important.

Acknowledgments

The work done in this chapter is in collaboration with Shabbir Ahmed, Santanu Dey, and Jean-Paul Watson. We would also like to thank Cynthia Phillips for helpful discussions during this work.

This work was supported by the Department of Energy, Office of Electricity Delivery & Energy Reliability, Advanced Grid Modeling Program led by Dr. Ali Ghassemian. Sandia National Laboratories is a multimission laboratory managed and operated by National Technology and Engineering Solutions of Sandia, LLC, a wholly owned subsidiary of Honeywell International, Inc., for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-NA0003525. SAND NO. 2021-5927 O. The views expressed in the article do not necessarily represent the views of the U.S. Department of Energy or the United States Government.

This work was performed under the auspices of the U.S. Department of Energy by Lawrence Livermore National Laboratory under Contract DE-AC52-07NA27344. This document was prepared as an account of work sponsored by an agency of the United States government. Neither the United States government nor Lawrence Livermore National Security, LLC, nor any of their employees makes any warranty, expressed or implied, or assumes any legal liability or responsibility for the accuracy, completeness, or usefulness of any information, apparatus, product, or process disclosed, or represents that its use would not infringe privately owned rights. Reference herein to any specific commercial product, process, or service by trade name, trademark, manufacturer, or otherwise does not necessarily constitute or imply its endorsement, recommendation, or favoring by the United States government or Lawrence Livermore National Security, LLC. The views and opinions of authors expressed herein do not necessarily state or reflect those of the United States government or Lawrence Livermore National Security, LLC, and shall not be used for advertising or product endorsement purposes.

CHAPTER 3

A SCALABLE LOWER BOUND FOR THE WORST-CASE RELAY ATTACK PROBLEM ON THE TRANSMISSION GRID

We consider a bilevel attacker-defender problem to find the worst-case attack on the relays that control transmission grid components. The attacker infiltrates some number of relays and renders all of the components connected to them inoperable, with the goal of maximizing load shed. The defender responds by minimizing the resulting load shed, redispatching using a DC Optimal Power Flow (DCOPF) problem on the remaining network. Though worst-case interdiction problems on the transmission grid have been studied for years, there remains a need for exact and scalable methods. Methods based on using duality on the inner problem rely on the bounds of the dual variables of the defender problem in order to reformulate the bilevel problem as a mixed-integer linear programming (MILP). Valid dual bounds tend to be large, resulting in weak linear programming relaxations and hence making the problem more difficult to solve at scale. Often smaller heuristic bounds are used, resulting in a lower bound. In this chapter we also consider a lower bound, where instead of bounding the dual variables, we drop the constraints corresponding to Ohm's law, relaxing DCOPF to capacitated network flow. We present theoretical results showing that, for uncongested networks, approximating DCOPF with network flow yields the same set of injections, and thus the same load shed, which suggests that this restriction likely gives a high-quality lower bound in the uncongested case. Furthermore, we show that in the network flow relaxation of the defender problem, the duals are bounded by 1, so we can solve our restriction exactly. Last, because the big-M values in the linearization are equal to 1 and network flow has a well-known structure, we see empirically that this formulation scales well computationally with increased network size. Through empirical experiments on 16 networks with up to 6,468 buses, we find that this bound is almost always as tight

as we can get from guessing the dual bounds, even for congested networks where the theoretical results do not hold. In addition, calculating the bound is approximately 150 times faster than achieving the same bound with the reformulation guessing the dual bounds.

3.1 Introduction

As the power grid becomes increasingly decentralized and networked, so does the potential for damaging cyberattacks. As is pointed out in Glenn et al. (2017), the United States electric grid was not originally designed to be networked, and control systems continue to become more complex. Both frequency and severity of cyberattacks on the grid have increased in the United States, and as smart grid capabilities continue to expand, distributed control of the grid could introduce additional vulnerabilities. Unlike physical attacks, which are more likely to affect a localized region of the grid, cyberattacks have the potential to infiltrate control centers, meaning an adversary could gain control over operations spanning large portions of the grid. For example, this was seen in the December 2015 cyberattack on Ukraine's power system, which affected about 225,000 customers. The attackers took control of an interface which let them open breakers, directly cutting power to customers. Simultaneously, they rendered the communication system useless with a denial of service attack (Sun et al., 2018). It is therefore increasingly critical to protect the grid against large and geographically disparate attacks.

In this chapter, we study a bilevel optimization problem which seeks to determine a worst-case cyberattack on the transmission grid. Such a model could be used to assess the vulnerability of the grid, and to decide portions of the grid which should be further protected, or perhaps air-gapped from the rest of the network. Additionally, this bilevel problem could be a subproblem in long-term planning problems such as physical and cyber network design problems seeking to minimize the risk of cyberattacks. The model we solve was originally presented in Castillo et al. (2019). We will refer to it as the *worst-case relay attack problem*. The attacker, seeking to maximize load shed, can choose a number of

relays to infiltrate, constrained by a budget parameter. All of the physical grid components these relays control are rendered inoperable, and the defender redispatches by solving a DCOPF model in order to minimize load shed.

Several prior works have considered a similar model, though in these models, the attacker targets grid components directly rather than relays. Note that the worst-case relay attack problem is no more difficult than interdiction problems which consider direct attacks on grid components. The problem of a worst-case grid attack was introduced by Salmeron et al. (2004a), and solved with a heuristic version of Benders decomposition. The subsequent work on methodology includes both exact and heuristic methods, but the exact methods rely on having strong bounds on the duals of the DCOPF linear program in order to scale in network size and in the size of the attacker's budget. Thus, it is common to use heuristic bounds on the dual variables in these methods, and solve the resulting restriction of the original problem.

Álvarez (2004) and Salmeron et al. (2004b) propose replacing the defender problem with its dual, linearizing the resulting bilinear terms, and solving the problem as a MILP. This technique only works for interdiction problems (that is, it only works on min-max problems), and for the method to be exact, the linearization requires valid bounds on the dual variables of the defender problem. Motto et al. (2005) adopt the same bilevel formulation, and reformulate the problem by leaving both the primal and dual variables of the inner problem in the model, generalizing the methodology beyond min-max problems and achieving better scalability in terms of attack budget. Arroyo (2010) considers a worst-case attack problem where only lines can be attacked, comparing the duality-based single-level reformulation with a mathematically equivalent reformulation using the Karush-Kuhn-Tucker (KKT) optimality conditions of the defender problem. Again, both of these methods are exact with valid bounds on the duals (or Lagrange multipliers). The author finds that, on a 24-bus system with attack budgets of up to 16 lines, the duality-

based method empirically outperforms the Karush-Kuhn-Tucker (KKT)-based method by a difference of several orders of magnitude.

Salmeron et al. (2009) present a generalized Benders decomposition algorithm based on the assumption that the total load shed cannot increase by more than the capacity of any one grid component when that component is attacked. This algorithm is capable of solving the problem on networks with more than 5,000 buses, but the scaling is not shown to accommodate increases in the size of the attack budget. For physical attacks, limited attack budgets are likely realistic, but for cyberattacks, attackers are not limited by physical resources and could therefore be able to attack large portions of the grid that might not be geographically correlated. In addition, the method is only exact if the assumption holds, which is not necessarily true in congested networks.

Sundar, Coffrin, et al. (2018) consider a probabilistic version of the problem which they solve with an algorithm similar to the benders approach in Salmeron et al. (2009). They compare several formulations for power flow in the defender problem, including the network flow restriction we analyze in this chapter. Their computational study explicitly shows the boundaries of tractability in terms of both the network size and the attacker budget. Their approach scales to networks with up to 2,383 buses, and attack budgets of up to 5 components. Sundar, Misra, et al. (2019) consider yet another variation of the problem in which the attacks are assumed to be spatially or topologically correlated. With a similar benders approach, they are able to solve on networks with up to 240 buses and attack budgets up to 6 lines. Sundar, Misra, et al. (2021) revisit this model and develop a cut generation algorithm based on a penalty-based reformulation. In this methodology, the only bounds on the DCOPF duals needed are bounds on the dual variables corresponding to the thermal limit constraints. Some of these are fixed to 0 for lines that can never be at full capacity. The authors compare an exact version of their method, where the duals that are not fixed to 0 are bounded by the total load in the system, to a heuristic method where

Table 3.1: Summary of scalability of previous literature on worst-case attack problem in terms of number of buses in the network as well as cardinality in the attack budget.

Citation	Maximum Network Size	Maximum Attack Budget
Salmeron et al., 2004b	48 buses	24 components
Salmeron et al., 2004a	48 buses	40 components
Álvarez, 2004	24 buses	6 components
Motto et al., 2005	48 buses	40 components
Salmeron et al., 2009	> 5000 buses	18 components
Arroyo, 2010	24 buses	16 lines

they bound these duals by 1. The number of iterations required for the heuristic version to converge tends to be at least an order of magnitude less than the exact method.

In summary, most of the existing methodology requires valid bounds on the dual variables of the DCOPF linear program to be exact. Since these are large, the scalability of exact methods is limited in terms of the size of the network and the size of the attack budget, as is summarized in Table 3.1. The assumptions in these methods are symptoms of a broader problem in bilevel optimization: All methods of dualizing the inner problem in order to combine it with the outer problem require relatively tight upper bounds on the dual variables of the inner problem (Smith and Song, 2020). Though it is common to use heuristics to calculate big-M values with which to linearize the KKT conditions of the inner problem, Pineda and Morales (2018) show that these heuristics can fail, even for bilevel problems with linear programming leader and follower problems. Furthermore, Kleinert et al. (2020) show that verifying the correctness of big-M values in bilevel optimization is as hard as solving the original problem. They suggest that, if we choose to solve bilevel problems by reformulating the follower's problem using duality or its KKT conditions, then we will have to resort to problem-specific information in order to generate valid big-M values. Last, while methods such as covering decomposition from Israeli and Wood (2002) are both exact and applicable to this problem (and do not require a big-M), the cuts to block previously-generated attacks are included in the benders algorithm from Salmeron et al. (2009), implying that without enhancement, this is not a scalable approach.

Despite the fact that the scalability of the existing approaches for the worst-case attack model is limited, there has been continued interest in the literature in solving extensions of this model and more complicated models which include this model. Bienstock and Verma (2010) develop a problem-specific algorithm for a variation of the problem where the attacker minimizes the number of lines necessary to attack in order to achieve a pre-specified amount of load shed. In addition, the authors provide a novel model in which the attacker antagonistically modifies the resistances of the power lines. Further extensions include the addition of transmission line switching as an option for the defender in Delgadillo et al. (2010) and Zhao and Zeng (2013), inclusion of both short- and medium-term impacts of attacks in Y. Wang and Baldick (2014), modeling attacks which unfold over time in Sayyadipour et al. (2016), modeling coordinated cyber and physical attacks in Li et al. (2016), and, as previously mentioned, adding the assumption of spatially correlated physical attacks in Sundar, Misra, et al. (2019) and Sundar, Misra, et al. (2021). In addition, there has been interest in trilevel planning problems such as defensive hardening of the network in Yuan et al. (2014), Alguacil et al. (2014), and Wu and Conejo (2017) and optimal design of the SCADA control system in Arguello et al. (2021).

In this chapter, we revisit the network flow restriction from Sundar, Coffrin, et al. (2018). That is, instead of solving DCOPF in the defender problem, we drop the Ohm's law constraints, simplifying the inner problem to capacitated network flow. This is a restriction of the original problem, as it expands the defender's feasible region, thus restricting the attacker's options. Applying it to get a lower bound for the worst-case relay attack problem, we show it can be used on networks with more than 6,000 buses with attack budgets ranging from small numbers of relays up to 30% of the network, enough to shed all of the load. While such an approach only gives a lower bound, we formally show that, when line capacities are large enough, the optimal objective value of the network flow restriction is the same as that of the original worst-case relay attack problem. This is because network flow is a good approximation of DCOPF when both formulations are projected into the space

of injections. That is, the line flows in the optimal solution of the network flow restriction may be dramatically different from those of DCOPF, but the load shed and generator dispatch will be the same. Since the formulation measures the severity of the attack in terms of load shed, the accuracy of the line flows will not effect the attack solution unless the network is congested. While we do expect the attacker to take advantage of his ability to create congestion, we find empirically that, even on congested instances, the bound we get from the network flow restriction is almost always as tight as we can find when we solve the original problem reformulated with improvised dual bounds.

As is observed in Roald and Molzahn (2019), in DCOPF, very few line limit constraints are ever tight, even accounting for variation in both demand and generation costs. In other words, in practice, transmission networks are rarely congested. Thus, it is not unexpected that the network flow restriction bound appears to be high quality. In addition, we find that we can obtain this bound within 20 minutes, even on large-scale networks with difficult-to-solve attack budgets. Though this is likely because capacitated network flow is a familiar and highly-optimized problem for commercial solvers, it is worth mentioning that network flow interdiction is itself a well-studied problem with some promising theoretical results which might eventually be applied to solve the network flow restriction. For example, Chestnut and Zenklusen (2017) give an approximation algorithm for an interdiction problem where the attacker eliminates edges in order to minimize the maximum s-t flow. With slight modifications (i.e., modeling generators and loads with mock lines to a super source and super sink respectively), the network flow restriction can be modeled as a maximum s-t flow, so Chestnut and Zenklusen (2017) and other combinatorial methods are applicable to it.

In summary, our contributions are:

1. A theoretical analysis of the network flow lower bound showing its quality on uncongested networks, and

2. A computational study on 16 networks of various sizes and levels of congestion, showing both that the network flow lower bound scales well computationally and that the quality of the bound is comparable to that of methods using heuristic bounds on the dual variables of DCOPF, even for congested networks where the theoretical results do not hold.

In the remainder of this chapter, we introduce the worst-case relay attack model in Section 3.2, introduce the network flow restriction in Section 3.3, state the main theoretical results related to it in Section 3.4, present a computational study demonstrating its efficacy in Section 3.5, and provide concluding thoughts in Section 3.7.

3.2 Worst-Case Relay Attack Formulation

The bilevel model is as follows:

$$\max_{\delta,u,v,w} \min_{f,p,\ell,\theta} \quad \sum_{s\in\mathcal{S}} \ell_s \tag{3.1a}$$

$$\text{s.t.} \quad \sum_{r\in\mathcal{R}} \delta_r \le U \tag{3.1b}$$

$$\sum_{r\in\mathcal{R}_l} \delta_r \ge 1 - v_l \qquad \forall l \in \mathcal{L} \tag{3.1c}$$

$$\sum_{r\in\mathcal{R}_g} \delta_r \ge 1 - w_g \qquad \forall g \in \mathcal{G} \tag{3.1d}$$

$$\sum_{r\in\mathcal{R}_s} \delta_r \ge 1 - u_s \qquad \forall s \in \mathcal{S} \tag{3.1e}$$

$$\delta_r \le 1 - v_l \qquad \forall l \in \mathcal{L}, r \in \mathcal{R}_l \tag{3.1f}$$

$$\delta_r \le 1 - w_g \qquad \forall g \in \mathcal{G}, r \in \mathcal{R}_g \tag{3.1g}$$

$$\delta_r \le 1 - u_s \qquad \forall s \in \mathcal{S}, r \in \mathcal{R}_s \tag{3.1h}$$

$$\delta_r \in \{0,1\} \qquad \forall r \in \mathcal{R} \tag{3.1i}$$

$$v_l \in \{0,1\} \qquad \forall l \in \mathcal{L} \tag{3.1j}$$

$$w_g \in \{0,1\} \qquad \forall g \in \mathcal{G} \tag{3.1k}$$

$$u_s \in \{0,1\} \qquad \forall s \in \mathcal{S} \tag{3.1l}$$

$$f_l \leq B_l(\theta_{o(l)} - \theta_{d(l)}) + 2\pi B_l(1 - v_l) \qquad \forall l \in \mathcal{L} \tag{3.1m}$$

$$f_l \geq B_l(\theta_{o(l)} - \theta_{d(l)}) - 2\pi B_l(1 - v_l) \qquad \forall l \in \mathcal{L} \tag{3.1n}$$

$$\sum_{l \in \mathcal{L}_s^+} f_l - \sum_{l \in \mathcal{L}_s^-} f_l + \sum_{g \in \mathcal{G}_s} p_g + \ell_s = D_s \qquad \forall s \in \mathcal{S} \quad (\mu) \tag{3.1o}$$

$$D_s(1 - u_s) \leq \ell_s \leq D_s \qquad \forall s \in \mathcal{S} \quad (\varphi, \psi) \tag{3.1p}$$

$$-\overline{F}_l v_l \leq f_l \leq \overline{F}_l v_l \qquad \forall l \in \mathcal{L} \quad (\lambda^+, \lambda^-) \tag{3.1q}$$

$$0 \leq p_g \leq \overline{P}_g w_g \qquad \forall g \in \mathcal{G} \quad (\gamma) \tag{3.1r}$$

$$\ell_s \geq 0 \qquad \forall s \in \mathcal{S} \tag{3.1s}$$

$$-\pi \leq \theta_s \leq \pi \qquad \forall s \in \mathcal{S} \tag{3.1t}$$

The attacker maximizes the total load shed and the defender minimizes it in (3.1a). Constraint (3.1b) ensures that the attacker does not exceed the cardinality budget for the number of relays he can attack. Constraints (3.1c)-(3.1e) enforce that if a component is unavailable, a relay which connects to it must have been attacked. The following three sets of constraints (3.1f)-(3.1h) enforce that if a relay connected to a line, generator, or load (respectively) is attacked, that component is unavailable to the defender. Constraints (3.1i)-(3.1l) give the domain of the attacker's variables. The defender's feasible region is defined by constraints (3.1m)-(3.1t). Constraints (3.1m)-(3.1n) represent Ohm's law when $v_l = 1$ and are trivial when $v_l = 0$. Constraint (3.1o) enforces power balance at each node. Constraints (3.1p)-(3.1t) enforce variable bounds and turn off components which are unavailable as a result of the attack. Note that we assume that the generator dispatch lower bound is 0. We do this to ensure that the defender problem is feasible for all attacks, since it is always feasible to generate no power and shed all the load.

3.3 Network Flow Restriction

We propose solving a restriction of problem (3.1) in which we drop constraints (3.1m) and (3.1n), which consequently removes the phase angle variables θ. That is, we propose solving:

$$\begin{aligned} \max_{\delta,u,v,w} \min_{f,p,\ell,\theta} \quad & \sum_{s\in\mathcal{S}} \ell_s \\ \text{s.t.} \quad & \text{(3.1b)-(3.1l), (3.1o)-(3.1s).} \end{aligned} \tag{3.2}$$

Note that (3.2) gives a lower bound to problem (3.1) since it expands the feasible region of the defender, giving him more options to respond to the attack, and therefore decreasing the load shed from the attack. We formulate a single-level bilinear reformulation of (3.2) by taking the dual of the defender problem:

$$\max_{\delta,u,v,w,\varphi,\psi,\lambda^+,\lambda^-,\gamma,\mu} \quad \begin{aligned} & -\sum_{l\in\mathcal{L}} \overline{F}_l(v_l\lambda_l^+ + v_l\lambda_l^-) - \sum_{g\in\mathcal{G}} \overline{P}_g w_g \gamma_g \\ & + \sum_{s\in\mathcal{S}} D_s((1-u_s)\varphi_s + \mu_s - \psi_s) \end{aligned} \tag{3.3a}$$

$$\text{s.t.} \quad \text{(3.1b)-(3.1l)}$$

$$\lambda_l^+ - \lambda_l^- + \mu_{d(l)} - \mu_{o(l)} = 0 \qquad \forall l \in \mathcal{L} \quad (f) \tag{3.3b}$$

$$\mu_{b(g)} - \gamma_g \leq 0 \qquad \forall g \in \mathcal{G} \quad (p) \tag{3.3c}$$

$$\varphi_s + \mu_s - \psi_s \leq 1 \qquad \forall s \in \mathcal{S} \quad (\ell) \tag{3.3d}$$

$$\varphi_s \geq 0 \qquad \forall s \in \mathcal{S} \tag{3.3e}$$

$$\psi_s \geq 0 \qquad \forall s \in \mathcal{S} \tag{3.3f}$$

$$\lambda_l^+ \geq 0 \qquad \forall l \in \mathcal{L} \tag{3.3g}$$

$$\lambda_l^- \geq 0 \qquad \forall l \in \mathcal{L} \tag{3.3h}$$

$$\gamma_g \geq 0 \qquad \forall g \in \mathcal{G}. \tag{3.3i}$$

Note that the objective function is bilinear, but since all the bilinear terms are products of a binary and a non-negative continuous variable, it is easily linearized if we have bounds on the continuous variables. We can show that for this problem, 1 is a valid upper bound for all the dual variables.

Observation 1. *In the formulation (3.3a)-(3.3i), it is valid to add the constraints*

$$-1 \leq \mu_s \leq 1 \qquad \forall s \in \mathcal{S} \tag{3.4}$$

$$\varphi_s \leq 1 \qquad \forall s \in \mathcal{S} \tag{3.5}$$

$$\psi_s \leq 1 \qquad \forall s \in \mathcal{S} \tag{3.6}$$

$$\lambda_l^+ \leq 1 \qquad \forall l \in \mathcal{L} \tag{3.7}$$

$$\lambda_l^- \leq 1 \qquad \forall l \in \mathcal{L} \tag{3.8}$$

$$\gamma_g \leq 1 \qquad \forall g \in \mathcal{G}. \tag{3.9}$$

The proof of Observation 1 relies on both the fact that, after the constraints corresponding to Ohm's law are removed, the inner problem's constraint matrix is totally unimodular and the fact that all the coefficients of the objective are 1. Note that without the latter property, our results may not hold. We give a formal proof of Observation 1 in Appendix F, section F.1. The application of the observation yields a mixed integer linear reformulation of (3.3):

$$\max_{\delta,u,v,w,\varphi,\psi,\lambda^+,\lambda^-,\gamma,\mu,\overline{\lambda}^+,\overline{\lambda}^-,\overline{\gamma},\overline{\varphi}} \quad \begin{aligned} &-\sum_{l\in\mathcal{L}} \overline{F}_l(\overline{\lambda}_l^+ + \overline{\lambda}_l^-) \\ &-\sum_{g\in\mathcal{G}} \overline{P}_g\overline{\gamma}_g + \sum_{s\in\mathcal{S}} D_s(\overline{\varphi}_s + \mu_s - \psi_s) \end{aligned} \tag{3.10a}$$

$$\text{s.t.} \quad \text{(3.1b)-(3.1l), (3.3b)-(3.3i)}$$

$$\overline{\lambda}_l^+ \leq v_l \qquad \forall l \in \mathcal{L} \tag{3.10b}$$

$$\lambda_l^+ - 1 + v_l \leq \overline{\lambda}_l^+ \leq \lambda_l^+ \qquad \forall l \in \mathcal{L} \tag{3.10c}$$

$$\overline{\lambda}_l^- \leq v_l \qquad \forall l \in \mathcal{L} \quad (3.10d)$$

$$\lambda_l^- - 1 + v_l \leq \overline{\lambda}_l^- \leq \lambda_l^- \qquad \forall l \in \mathcal{L} \quad (3.10e)$$

$$\overline{\gamma}_g \leq w_g \qquad \forall g \in \mathcal{G} \quad (3.10f)$$

$$\gamma_g - 1 + w_g \leq \overline{\gamma}_g \leq \gamma_g \qquad \forall g \in \mathcal{G} \quad (3.10g)$$

$$\overline{\varphi}_s \leq 1 - u_s \qquad \forall s \in \mathcal{S} \quad (3.10h)$$

$$\varphi_s - u_s \leq \overline{\varphi}_s \leq \varphi_s \qquad \forall s \in \mathcal{S} \quad (3.10i)$$

$$\overline{\lambda}_l^+ \geq 0 \qquad \forall l \in \mathcal{L} \quad (3.10j)$$

$$\overline{\lambda}_l^- \geq 0 \qquad \forall l \in \mathcal{L} \quad (3.10k)$$

$$\overline{\gamma}_g \geq 0 \qquad \forall g \in \mathcal{G} \quad (3.10l)$$

$$\overline{\varphi}_s \geq 0 \qquad \forall s \in \mathcal{S}. \quad (3.10m)$$

Note that the optimal value of (3.10) is a lower bound to that of (3.1).

Suppose z^* is the optimal objective function value of (3.10) and let $(\delta^*, u^*, v^*, w^*)$ be an optimal solution of (3.10) corresponding to the attacker's variables. Since we have removed the constraints corresponding to Ohm's law, it is possible that if we fix the attacker variables (δ, u, v, w) to $(\delta^*, u^*, v^*, w^*)$, the defender has no DCOPF-feasible solution. In particular, this case implies that fixing (δ, u, v, w) to $(\delta^*, u^*, v^*, w^*)$ will lead to a higher load shed than z^*. Thus, in order to (i) obtain a feasible solution to our original problem (3.1) and (ii) possibly improve the bound obtained from (3.10), we apply the steps described in Algorithm 1.

Algorithm 1: Network Flow Lower Bound

Input: All parameters of the power network, relays, and budget
Output: A high quality feasible solution to (3.1).

1 Solve (3.10). Let $(\delta^*, u^*, v^*, w^*)$ be optimal solution of (3.10) corresponding to the attacker's variables.
2 Fix (δ, u, v, w) to $(\delta^*, u^*, v^*, w^*)$ and solve the resulting defender's DCOPF problem (with Ohm's law).
3 Return $(\delta^*, u^*, v^*, w^*)$, and the solution to the DCOPF problem.

In the remainder of the chapter, we call the the resulting lower bound obtained from the load shed corresponding to the solution returned by Algorithm 1 the *network flow lower bound (NFLB)*.

In Section 3.5, we will show that empirically we find that (3.10) is efficiently solvable, even for large networks, and, as far as can be measured, NFLB is a high-quality lower bound. In the following section, we give some theoretical results which provide intuition for the good quality of NFLB.

3.4 Theoretical Analysis of the Quality of the Network Flow Restriction

Though there are various notions of congestion in power networks, in this chapter we describe a network as congested when the thermal limits on the transmission lines prevent a solution with less load shed. Note that, in a DCOPF model, a bound on the phase angle difference can also be a source of congestion, but for a line l, this is only the case when

$$2B_l\Theta < \overline{F}_l, \tag{3.11}$$

where Θ is the phase angle difference bound. That is, phase angles will become the limiting factor in how much power can be moved through the network if the maximum phase angle difference multiplied by the susceptance provides a tighter bound on the line flow than the thermal limit does. However, in the case that (3.11) is true, we can replace the thermal limit with the left-hand side of (3.11) and drop phase angle difference bounds from the problem. Thus, without loss of generality, in this chapter we model DCOPF without phase angle difference bounds and consider congestion to be caused by restrictive thermal limits.

In the following, we show that, when the thermal limits are sufficiently large, it is always possible to find a DCOPF solution with the same injections as a network flow solution on the same network. Throughout this section, we will assume that there is exactly one generator per bus. Buses which do not have a generator can be represented as having a

generator with maximum capacity 0, and buses with multiple generators can be represented as having one generator with capacity set to the sum of the capacities of the originals. This is again because we assume that the minimum dispatch for a generator is always 0. For notational convenience, we first introduce some definitions.

Definition 1. *Represent the network as a digraph $G(\mathcal{S}, \mathcal{L})$ and let $N \in \{0, 1, -1\}^{|\mathcal{S}|\times|\mathcal{L}|}$ be the node-arc incidence matrix (where buses are nodes and lines are arcs). Let $\{d_s\}_{s\in\mathcal{S}}$ be a set of injections such that $\sum_{s\in\mathcal{S}} d_s = 0$[1]. Then we say:*

- *The injection vector d is flow-polytope feasible if there exists a vector of flows that satisfies thermal limits and flow conservation given the nodal injection values, i.e., d is flow-polytope feasible if there exists $f \in \mathbb{R}^{|\mathcal{L}|}$ such that*

$$Nf = d, \text{ and } |f_l| \leq \overline{F}_l, \ \forall l \in \mathcal{L}.$$

- *The injection vector d is DCOPF feasible if there exists a flow vector that satisfies thermal limits, flow conservation given the nodal injection values, and Ohm's law. That is, d is DCOPF feasible if there exists $f \in \mathbb{R}^{|\mathcal{L}|}$ such that*

$$Nf = d, |f_l| \leq \overline{F}_l, \ \forall l \in \mathcal{L}, \text{ and } \exists\, \theta \in \mathbb{R}^{|\mathcal{S}|} \text{ such that } f_l = B_l(\theta_{o(l)} - \theta_{d(l)}), \ \forall l \in \mathcal{L}.$$

Note that we do not require that θ satisfy the bounds given in (3.1t).

The set of DCOPF feasible injections is contained in the set of flow-polytope feasible injections. We will show that the reverse is also true for uncongested networks, that is, when the thermal limits are sufficiently large.

Definition 2. *Given a connected digraph $G(\mathcal{S}, \mathcal{L})$, consider a partition of the nodes formed by removing all the cut-arcs in the underlying graph and labeling the sets of nodes in each*

[1] In terms of the notation used to describe (3.1), for a bus s, $d_s = \sum_{g\in\mathcal{G}_s} p_g - (D_s - \ell_s)$, i.e., the power generated at the bus minus the load served at the bus.

of the resulting connected components as $V^1, V^2, \ldots, V^m$. Let

$$r(G) := \max_{1 \leq i \leq m} |V^i|.$$

Let $E(V^i)$ be the set of arcs with both end points in V^i. We will call the set of arcs $\cup_{i=1}^m E(V^i)$ non-cut-arcs.

Note that, in power networks, a partition of the nodes into more than one non-empty set is unusual since a cut-arc represents a single point of failure. Thus we expect that for many of these networks, $r(G) = |\mathcal{S}|$. Intuitively, the non-cut-arcs are the only ones for which the thermal limits could restrict our ability to find DCOPF feasible flows for a set of injections. This is because, on a tree (and in the absence of phase angle bounds), there always exist phase angles such that a flow-polytope feasible flow is also DCOPF feasible. Thus, the injections are certainly feasible. Stated more simply, Ohm's law poses no additional restriction on flows if there are no cycles in the network. Thus, our notion of "large enough" thermal limits only applies to arcs which appear in cycles. Formally, we have the following theorem, which we will prove in Appendix F, section F.2:

Theorem 1. *Consider a DCOPF problem on a connected digraph $G(\mathcal{S}, \mathcal{L})$. Let $\mathcal{S} = V^1 \cup V^2 \cup \cdots \cup V^m$ such that $V^i \cap V^j = \emptyset$ for all $i \neq j$, and when we contract the nodes in V^i into one node, the resulting graph is a tree. Let $r(G) := \max_{1 \leq i \leq m} |V^i|$. (Note that we can always select $r(G) = |\mathcal{S}|$.) Recall $\overline{F}_l$ is the thermal limit on arc l. Let $B_{\max}$ and $B_{\min}$ be the maximum and minimum susceptance respectively. If $d \in \mathbb{R}^{|\mathcal{S}|}$ is flow-polytope feasible and*

$$\overline{F}_l \geq \sqrt{\frac{B_{\max}}{B_{\min}}} \frac{\sqrt{r(G) - 1}}{2} \|d\|_1 \; \forall l \in \cup_{i=1}^m E(V^i), \tag{3.12}$$

then d is also a DCOPF feasible injection.

Essentially, this means that, in uncongested networks, network flow is a good approximation for DCOPF when we consider the space of feasible injections. More precisely:

Corollary 1. *Consider the following problem:*

$$
\begin{aligned}
z^* = \min_{\ell,f,p,\theta} \quad & \sum_{s\in\mathcal{S}} \ell_s \\
\textit{s.t.} \quad & \sum_{l\in\mathcal{L}_s^+} f_l - \sum_{l\in\mathcal{L}_s^-} f_l + \sum_{g\in\mathcal{G}_s} p_g + \ell_s = D_s \quad \forall s \in \mathcal{S} \\
& f_l = B_l(\theta_{o(l)} - \theta_{d(l)}) \qquad \forall l \in \mathcal{L} \\
& -\overline{F}_l \le f_l \le \overline{F}_l \qquad \forall l \in \mathcal{L} \\
& 0 \le \ell_s \le D_s \qquad \forall s \in \mathcal{S} \\
& 0 \le p_g \le \overline{P}_g \qquad \forall g \in \mathcal{G}
\end{aligned}
\tag{3.13}
$$

and its relaxation

$$
\begin{aligned}
z^l = \min_{\ell,f,p} \quad & \sum_{s\in\mathcal{S}} \ell_s \\
\textit{s.t.} \quad & \sum_{l\in\mathcal{L}_s^+} f_l - \sum_{l\in\mathcal{L}_s^-} f_l + \sum_{g\in\mathcal{G}_s} p_g + \ell_s = D_s \quad \forall s \in \mathcal{S} \\
& -\overline{F}_l \le f_l \le \overline{F}_l \qquad \forall l \in \mathcal{L} \\
& 0 \le \ell_s \le D_s \qquad \forall s \in \mathcal{S} \\
& 0 \le p_g \le \overline{P}_g \qquad \forall g \in \mathcal{G}
\end{aligned}
\tag{3.14}
$$

Let $G(\mathcal{S},\mathcal{L})$ be the network digraph, $r(G)$ be as defined in Theorem 1, and $D \in \mathbb{R}^{|\mathcal{S}|}$ be the vector of values D_s. Then if $\overline{F}_l \ge \sqrt{\frac{B_{\max}}{B_{\min}}}\sqrt{r(G)-1}\|D\|_1$, we have $z^* = z^l$.

We provide a proof of this result in Appendix F, section F.3. It follows that, when the thermal limits respect the bound given in Corollary 1 and we disregard phase angle bounds, the network flow relaxation of the worst-case relay attack problem is tight.

3.4.1 Tightness of the Bound from Theorem 1

As we will show through our empirical study in Section 3.5, we believe that in practice, the bound on the thermal limits in Theorem 1 is quite conservative. That is, even for thermal

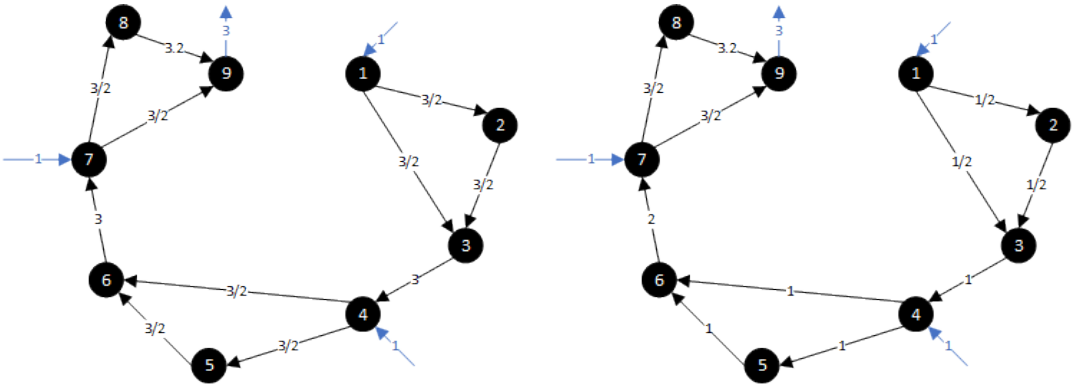


Figure 3.1: Example of artificial digraph used in the proof of Proposition 2 with $n = 3$. Black nodes and arcs represent network structure and blue arcs represent injections. The arcs are labeled with the thermal limits in the left-hand image and with flow-polytope feasible flows in the right-hand image.

limit values much smaller than the bound given in Theorem 1, NFLB is the same as the optimal value of the original problem. However, we can show that for artificial instances the result is tight within a constant:

Proposition 2. *There exists a constant c and a family of digraphs $\{G^n(V^n, E^n)\}_{n\in\mathbb{N}}$ where all susceptances are equal to 1 with corresponding injections $d^n \in \mathbb{R}^{|V^n|}$ and $\lim_{n\to\infty} \|d^n\|_1 = \infty$ such that if*

$$\overline{F}_e < c\sqrt{\frac{B_{\max}}{B_{\min}}}\frac{\sqrt{r(G)-1}}{2}\|d^n\|_1$$

for all non-cut-arcs e then d^n is a flow-polytope feasible injection but not a DCOPF feasible injection.

A formal proof is provided in Appendix F, section F.4, but an example of such a digraph is shown in Figure 3.1. In essence, we can construct a digraph where the thermal limits satisfy the requirement of the theorem, but the triangles prevent there existing phase angles such that the lines can be used at capacity. In practice, such an instance would be surprising, since most arcs in real networks are non-cut-arcs in order to prevent a single point of failure.

3.5 Computational Results

The current state-of-the-art. As discussed in the Section 3.1, the current state-of-the-art to obtain a lower bound for (3.1) is to solve single-level reformulations of (3.1). We present two single-level reformulations of (3.1) in Appendix E. In the first formulation, problem (E.1), we use logical constraints to give an exact reformulation of (3.1) because we do not specify upper bounds on the dual variables. It is possible to express this model using Gurobi's `IndicatorConstraints`. While this problem is exact, we will show later in this section that the computational time to solve it is prohibitively large. We therefore also consider a mixed integer linear programming reformulation of (3.1), given in (E.2), in which upper bounds on the dual variables (M) are used to linearize the implications in (E.1). Again, as discussed before, we typically do not have good knowledge of these bounds M, and thus solving (E.1) with a heuristic value of M leads to a lower bound.

Our Goal. Since the network flow restriction also provides a lower bound for the worst-case relay attack problem (i.e. NFLB), we seek to answer two questions in this section:

1. *Computational Tractability*: What is the computational advantage of solving the network flow restriction over approaches that solve the single-level MILP obtained using arbitrary/heuristic bounds on the dual variables?

2. *Quality:* What is the quality of the network flow restriction solutions, i.e., that of NFLB?

Question 1. In order to address Question 1, we compare the time it takes to obtain NFLB with the time it takes to find a solution with single-level reformulations of (3.1). In particular, after using Algorithm 1, we try to get a sense of the time it takes to find an equivalent-quality solution using the current state-of-the-art. Therefore we proceed as follows:

- We run Algorithm 1, which involves solving (3.10) followed by solving an instance of DCOPF. We record the time needed to run Algorithm 1.
- We have two choices for single-level reformulations of (3.1):
  - Solve (E.1): We cut off the run when the lower bound is equal to the NFLB or terminate after 4 hours if we did not achieve the NFLB before then. Unfortunately, while this method is the most attractive theoretically since problem (E.1) is an exact reformulation, we found that, even on the smallest test case, that it is much slower to solve than (E.2), taking over 4 hours to prove optimality on a 118-bus case with a 5% attack budget. For this reason, we only report results with this model where we cut off at NFLB.
  - Solve (E.2): We first need to decide M values. We set the value of M in (E.2) to be the ceiling of the largest dual variable of the linear program solved in line 2 of Algorithm 1. For all of our test networks, applying the above procedure, we found $M = 2$ or $M = 3$. It is possible we have discarded better solutions with this choice of M, but we at least ensure that we do not cut off the solution we have already found. In addition, in experiments not reported here, we attempted larger values of M for the smaller cases and found that the solution time scales badly with increases in M. Furthermore, we still did not find better solutions than the one corresponding to our chosen value of M. Again, we cut off the run of solving (E.2) when the lower bound is equal to the NFLB or terminate after 4 hours if we did not achieve the NFLB before then. We use Gurobi to solve (E.2), setting Gurobi's `MIPFocus` parameter to 1 to prioritize finding good quality solutions.

Question 2. In order to answer Question 2, ideally, we should compare NFLB to the optimal solution of (3.1). This is difficult to answer since we do not know of a non-trivial upper bound for the worst-case relay attack problem. In theory, we could accomplish this

Table 3.2: Details of test instances used in the computational study. The last two columns give insight into congestion. We solve DCOPF with no attack and give the percentage of lines operating at their limits in the 'Percentage of Thermal Limits Tight' column and the percentage of buses with a phase angle at its bound in the 'Percentage of Phase Angle Bounds Tight' column.

Instance	Number of Buses	Number of Lines	Number of Generators	Percentage of Thermal Limits Tight	Percentage of Phase Angle Bounds Tight
118Blumsack	118	186	19	0.54%	1.70%
300Kocuk	300	411	61	2.92%	0.67%
500tamu	500	597	90	0.00%	1.40%
1354pegase	1354	1991	260	1.21%	0.74%
1354pegase_api	1354	1991	260	0.55%	1.11%
1354pegase_sad	1354	1991	260	1.31%	0.59%
1888rte	1888	2531	297	0.91%	0.05%
1888rte_api	1888	2531	297	2.96%	0.05%
1888rte_sad	1888	2531	297	1.07%	0.05%
1951rte	1951	2596	391	0.23%	0.05%
1951rte_api	1951	2596	391	4.08%	0.05%
1951rte_sad	1951	2596	391	0.58%	0.05%
2848rte	2848	3776	547	0.66%	0.04%
3012wp	3012	3572	502	0.03%	0.03%
3375wp	3374	4161	596	0.26%	0.03%
6468rte	6468	9000	1295	0.11%	0.02%

by solving (E.1). However, as mentioned earlier, we found that solving (E.1) using Gurobi's `IndicatorConstraints` does not scale well enough to beat NFLB. We therefore approach this question by comparing to the best lower bound obtained from the single-level reformulation of (E.2) with the heuristic choice of M described above, warmstarted with the solution we found in Algorithm 1, and given a time limit of 4 hours.

Software and Hardware Specification. Our models are implemented in Pyomo (Bynum et al., 2021; Hart et al., 2011) using Gurobi 9.0.2 as the solver (Gurobi Optimization, LLC, 2020). The experiments are run giving Gurobi 8 threads on a server with 40 Intel Xeon 2.20GHz CPUs and 256GB of RAM.

3.5.1 Test Networks

We present results on 16 different networks ranging from 118 buses to 6,468 buses and with varying levels of congestion. Details of the networks are given in Table 3.2. Note that

118Blumsack is the IEEE 118 bus network as modified in Blumsack et al. (2007). This is a very congested network, which we use intentionally since congestion can break down the assumptions on dual bounds used in prior work and also renders our theoretical guarantees moot. The 300Kocuk case is the IEEE 300 bus case, as modified in Kocuk, Jeon, et al. (2016). It also has been modified to be more congested than the original. The other cases are used as they are presented in Babaeinejadsarookolaee et al. (2019). Note that the cases with names ending in '_api' and '_sad' are congested modifications of the instance which shares the prefix of their name.

Since these test networks do not include any information about the control systems, for the sake of demonstration, we assume that there is one relay per bus which controls that bus, all the generators at that bus, and all the lines adjacent to the bus. For each of the test networks, we find a solution for the worst-case relay attack problem for attacker budgets of 1%, 3%, 5%, 7%, 10%, 13%, 15%, 20%, 25%, and 30% of the relays in the grid.

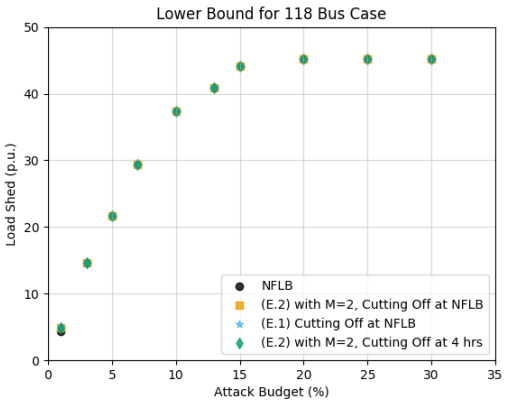
3.5.2 Network Flow Restriction Results

Difficulty in solving instances as a function of budget. Both very small budgets and very large budgets turn out to be easier problems for Gurobi. As is also observed in Bienstock and Verma (2010), we typically see that the computational time is longest for mid-range budgets. This is intuitive since for small budgets there are fewer possible attacks, and for larger budgets, the attacker is able to shed all the load in the system, so the trade-offs are no longer interesting. This concept is illustrated for a couple of the test networks in Figure 3.2. In Figures 3.2a and 3.2c, we see that as the attack budget increases, the amount of load shed achievable by the attacker increases and eventually saturates at the total load in the system. Note that these plot the lower bound achieved at the 4-hour time limit, explaining why (E.1) can have a lower objective value than the other models. In Figure 3.2b, we see that the most difficult problems computationally are at the elbow of the curve in

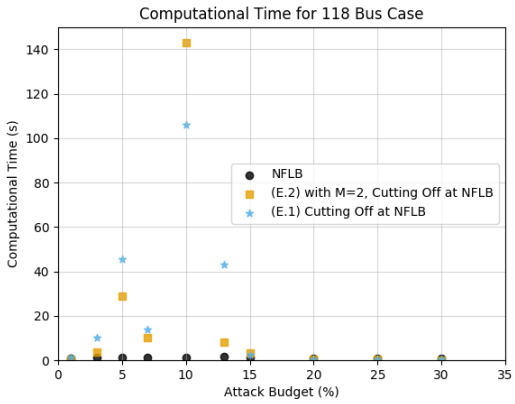
Figure 3.2a. Similarly in Figure 3.2d, for a larger case, all the small budgets are difficult for Gurobi, but the problem becomes trivial after passing the saturation point.

Results. The results from the network flow restriction and the experiments on problems (E.1) and (E.2) are shown in Tables 3.3, 3.4, 3.5, and 3.6 for the twelve smaller test instances. The first column shows the attack budget as a percentage of the relays in the system. The second column translates this into an integer number of relays which can be attacked, that is, the value of U for that instance. The third column gives the best known lower bound from among all our experiments. This is the highest known load shed the attacker can achieve, given in per unit[2]. In the "NFLB" columns, "Quality" is the load shed from the attack found by Algorithm 1 as a percentage of the best known lower bound. The time in seconds that it takes to run Algorithm 1 is reported in the "Time" column. In the next three columns, we report results related to Question 2 above, that is, determining the quality of NFLB. Recall that, in this experiment, we solve (E.2) using the heuristic M, warmstarting with the solution corresponding to NFLB, and allowing Gurobi a 4-hour time limit. The "Problem (E.2) Quality" column gives the load shed this experiment achieved as a percentage of the best known lower bound. The "Problem (E.2) Time" column gives the time for the Gurobi solve. The "Problem (E.2) Gap" column reports the gap after the 4 hours. Note that this is not a gap with a valid upper bound for the worst-case relay attack problem, but is instead a measure of how close Gurobi was to proving optimality on the particular restriction it was solving, in this case with the dual variables bounded by 2 for all but the 1888rte_api and 1951rte_api cases, where the dual variables are bounded by 3. That is, Gurobi's upper bound is a bound on the best feasible solution achievable with this restriction. In the last four columns, we report results related to Question 1 from the beginning of this section, in which we compare to solving the worst-case relay attack problem using formulations from prior literature. In these experiments, we do not warmstart the

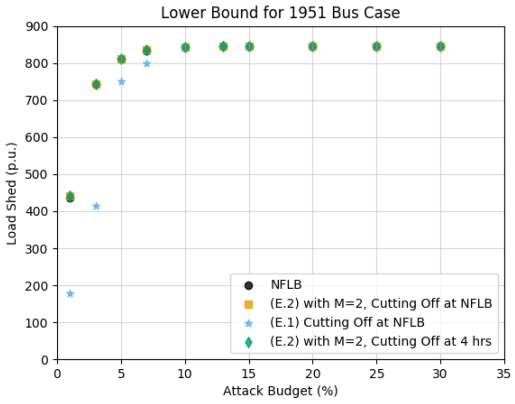
[2]As is typical in power systems modeling, in order to have better-scaled models, we quantify power "per unit," that is, in 100 MW units.



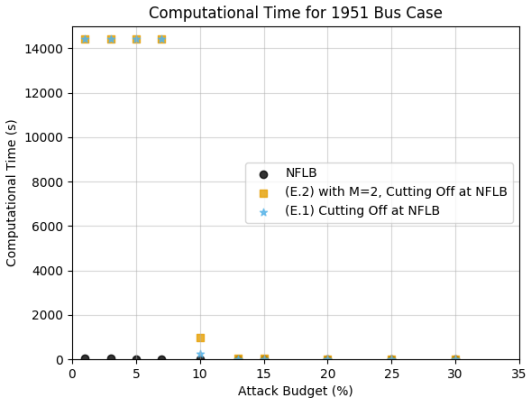
(a) Lower bounds for 118 Bus Case.



(b) Computational times for 118 Bus Case.



(c) Lower bounds for 1951 Bus Case.



(d) Computational times for 1951 Bus Case.

Figure 3.2: Computational times and bounds for different attack budgets on the 118Blumsack and 1951rte cases. NFLB is obtained by running Algorithm 1 using Gurobi. The other bounds are obtained using Gurobi to solve (E.1) and (E.2) terminating either when the bound reaches NFLB or after 4 hours. The numbers visualized in plots 3.2b and 3.2d are given in the "NFLB Time," "Problem (E.2) Time to NFLB", and "Problem (E.1) Time to NFLB" columns of Tables 3.3 and 3.4.

Gurobi solves, and we cut off the solve when Gurobi achieves NFLB, if that is before the time limit of 4 hours. We report the load shed achieved as a percentage of the best known lower bound as well as the time it takes Gurobi to find a solution whose objective value is as good as NFLB when solving (E.2) and (E.1) respectively. As mentioned previously, we do not report results where we continue solving (E.1) after it achieves NFLB because we found it slow to find a solution as good as that obtained by Algorithm 1, even for the smaller test cases.

Gurobi hits the 4-hour time limit consistently for the more difficult budgets in the larger of these instances (i.e., Gurobi does not reach the NFLB within 4 hours). Therefore we did not compare with solving either (E.1) or (E.2) for the four largest instances, and instead report the results of just the network flow restriction in Table 3.7. Column "NFLB" gives the load shed from the attack found by Algorithm 1 in per unit, and "NFLB Time" gives the time taken to run Algorithm 1.

Table 3.3: Network flow restriction results on the six smallest cases, part 1. For each instance, we show results for 10 different budgets for the percentage of relays that can be attacked. The best known achievable load shed is in the "Best Known LB" column. In the following two columns we give NFLB as a percentage of the best known solution for the instance and the computational time for Algorithm 1. The next three columns show the load shed attained by the solution we get from running (E.2) for up to 4 hours, the running time, and Gurobi's optimality gap at termination. The last four columns show the quality of the solution achieved and the times for Gurobi to achieve NFLB when solving problems (E.2) and (E.1) respectively.

Instance	Budget (%)	# of Relays	Best Known LB	**NFLB**		**Question 2**			**Question 1**			
				Quality	Time (s)	Problem (E.2) Quality	Problem (E.2) Time (s)	Problem (E.2) Gap	Problem (E.2) Quality	Problem (E.2) Time to NFLB (s)	Problem (E.1) Quality	Problem (E.1) Time to NFLB (s)
118Blumsack	1	1	4.93	89.25%	1.02	100.00%	1.30	0.00%	100.00%	0.32	100.00%	1.17
	3	4	14.63	100.00%	1.14	100.00%	64.97	0.00%	100.00%	3.70	100.00%	10.14
	5	6	21.72	100.00%	1.42	100.00%	720.08	0.00%	100.00%	28.83	100.00%	45.58
	7	8	29.38	100.00%	1.12	100.00%	387.48	0.00%	100.00%	10.24	100.00%	14.00
	10	12	37.31	100.00%	1.25	100.00%	1294.79	0.00%	100.00%	142.95	100.00%	105.88
	13	15	40.93	100.00%	1.71	100.00%	911.24	0.00%	100.00%	8.04	100.00%	43.11
	15	18	44.19	100.00%	1.16	100.00%	102.94	0.00%	100.00%	3.27	100.00%	2.53
	20	24	45.19	100.00%	0.80	100.00%	0.07	0.00%	100.00%	0.33	100.00%	0.35
	25	30	45.19	100.00%	0.84	100.00%	0.06	0.00%	100.00%	0.29	100.00%	0.31
	30	35	45.19	100.00%	0.80	100.00%	0.07	0.00%	100.00%	0.12	100.00%	0.29
300Kocuk	1	3	52.48	91.73%	2.26	100.00%	14.83	0.00%	100.00%	5.35	95.62%	15.98
	3	9	99.55	99.77%	3.71	100.00%	14400.01	22.75%	100.00%	99.53	100.00%	3065.81
	5	15	130.24	100.00%	6.47	100.00%	14400.02	27.94%	100.00%	385.57	89.58%	14400.00
	7	21	152.01	100.00%	8.84	100.00%	14400.01	28.92%	100.00%	538.82	97.95%	14400.01
	10	30	184.07	100.00%	5.90	100.00%	14400.01	19.99%	100.00%	199.14	98.35%	14400.00
	13	39	211.37	100.00%	3.62	100.00%	14400.01	9.16%	100.00%	355.72	99.71%	14400.01
	15	45	224.88	100.00%	3.18	100.00%	14400.01	4.27%	100.00%	212.62	99.63%	14400.00
	20	60	238.48	100.00%	2.35	100.00%	0.13	0.00%	100.00%	19.44	100.00%	4.10
	25	75	238.48	100.00%	2.07	100.00%	0.10	0.00%	100.00%	2.32	100.00%	1.72
	30	90	238.48	100.00%	1.59	100.00%	0.10	0.00%	100.00%	1.19	100.00%	2.74
500tamu	1	5	16.79	100.00%	13.71	100.00%	12764.73	0.00%	100.00%	3872.70	88.62%	14400.01
	3	15	71.88	100.00%	3.92	100.00%	14400.02	7.42%	100.00%	44.53	100.00%	82.53
	5	25	77.26	100.00%	2.88	100.00%	14400.02	0.32%	100.00%	44.11	100.00%	81.06
	7	35	77.51	100.00%	2.87	100.00%	0.15	0.00%	100.00%	14.38	100.00%	20.61
	10	50	77.51	100.00%	2.17	100.00%	0.15	0.00%	100.00%	2.82	100.00%	4.97
	13	65	77.51	100.00%	2.25	100.00%	0.15	0.00%	100.00%	2.16	100.00%	1.73
	15	75	77.51	100.00%	2.89	100.00%	0.15	0.00%	100.00%	1.02	100.00%	1.74
	20	100	77.51	100.00%	2.08	100.00%	0.15	0.00%	100.00%	3.09	100.00%	1.90
	25	125	77.51	100.00%	2.15	100.00%	0.15	0.00%	100.00%	1.66	100.00%	1.59
	30	150	77.51	100.00%	2.21	100.00%	0.15	0.00%	100.00%	0.99	100.00%	1.41

Table 3.4: Network flow restriction results on the six smallest cases, part 2. The columns are labeled as in Table 3.3. Note that, in the Question 1 results, in cases where problem (E.2) runs for 4 hours but has a quality of 100.00%, this is a symptom of rounding: The NFLB is not quite achieved within the time limit, but that is not reflected within the two decimal places in this table. Also note that the Question 1 experiment solving E.2 occasionally finds the best known solution since it can exceed the NFLB in the iteration before it terminates.

Instance	Budget (%)	# of Relays	Best Known LB	**NFLB**		**Question 2**			**Question 1**			
				Quality	Time (s)	Problem (E.2) Quality	Problem (E.2) Time (s)	Problem (E.2) Gap	Problem (E.2) Quality	Problem (E.2) Time to NFLB (s)	Problem (E.1) Quality	Problem (E.1) Time to NFLB (s)
1354pegase	1	14	231.67	100.00%	139.59	100.00%	14400.04	135.91%	97.74%	14400.02	47.40%	14400.01
	3	41	532.21	100.00%	44.77	100.00%	14400.03	30.74%	98.91%	14400.05	68.91%	14400.01
	5	68	653.14	100.00%	23.55	100.00%	14400.05	9.19%	99.18%	14400.06	93.28%	14400.01
	7	95	698.13	100.00%	18.53	100.00%	14400.18	6.21%	99.87%	14400.26	98.21%	14400.00
	10	135	735.70	100.00%	20.94	100.00%	14400.02	0.78%	100.00%	14400.03	99.69%	14400.01
	13	176	741.46	100.00%	12.55	100.00%	6.48	0.00%	100.00%	179.52	100.00%	77.53
	15	203	741.46	100.00%	9.30	100.00%	0.45	0.00%	100.00%	85.14	100.00%	64.13
	20	271	741.46	100.00%	7.03	100.00%	0.46	0.00%	99.99%	15.57	100.00%	12.61
	25	338	741.46	100.00%	6.23	100.00%	0.47	0.00%	100.00%	9.65	100.00%	8.50
	30	406	741.46	100.00%	6.10	100.00%	0.47	0.00%	100.00%	5.09	100.00%	8.91
1888rte	1	19	292.08	100.00%	45.70	100.00%	14400.03	76.57%	100.00%	8629.13	36.04%	14400.01
	3	57	525.44	100.00%	28.92	100.00%	14400.02	8.02%	100.00%	3790.54	85.52%	14400.05
	5	94	572.74	100.00%	27.58	100.00%	14400.17	2.45%	99.97%	14400.18	70.71%	14400.01
	7	132	591.76	100.00%	21.69	100.00%	14400.06	0.73%	99.99%	14400.07	97.80%	14400.02
	10	189	596.07	100.00%	12.60	100.00%	0.74	0.00%	100.00%	360.94	100.00%	28.14
	13	245	596.07	100.00%	9.42	100.00%	0.77	0.00%	100.00%	43.09	100.00%	23.44
	15	283	596.07	100.00%	8.83	100.00%	0.76	0.00%	100.00%	15.65	100.00%	16.64
	20	378	596.07	100.00%	8.38	100.00%	0.72	0.00%	100.00%	5.70	100.00%	14.25
	25	472	596.07	100.00%	8.12	100.00%	0.74	0.00%	100.00%	14.02	99.99%	8.88
	30	566	596.07	100.00%	8.06	100.00%	0.73	0.00%	100.00%	6.55	99.99%	9.77
1951rte	1	20	441.07	99.06%	42.70	99.97%	14400.02	65.13%	100.00%	14400.03	40.21%	14400.01
	3	59	743.17	100.00%	35.71	100.00%	14400.04	9.32%	99.89%	14400.04	55.82%	14400.03
	5	98	810.79	100.00%	26.35	100.00%	14400.20	2.51%	100.00%	14400.04	92.46%	14400.01
	7	137	834.51	100.00%	26.13	100.00%	14400.02	0.97%	100.00%	14400.06	95.85%	14400.01
	10	195	844.23	100.00%	18.37	100.00%	14.83	0.00%	100.00%	972.18	100.00%	255.64
	13	254	844.27	100.00%	13.43	100.00%	2.51	0.00%	100.00%	47.39	100.00%	15.35
	15	293	844.27	100.00%	11.70	100.00%	0.80	0.00%	100.00%	44.41	100.00%	14.70
	20	390	844.27	100.00%	11.21	100.00%	0.91	0.00%	100.00%	6.63	100.00%	14.59
	25	488	844.27	100.00%	11.19	100.00%	3.26	0.00%	100.00%	10.19	100.00%	11.55
	30	585	844.27	100.00%	11.09	100.00%	2.16	0.00%	100.00%	9.46	100.00%	12.47

Table 3.5: Network flow restriction results on the 'api' congested cases. The columns are labeled as in Table 3.3.

Instance	Budget (%)	# of Relays	Best Known LB	NFLB		Question 2			Question 1			
				Quality	Time (s)	Problem (E.2) Quality	Problem (E.2) Time (s)	Problem (E.2) Gap	Problem (E.2) Quality	Problem (E.2) Time to NFLB (s)	Problem (E.1) Quality	Problem (E.1) Time to NFLB (s)
1354pegase_api	1	14	223.18	100.00%	60.56	100.00%	14400.03	109.28%	100.00%	14400.04	48.75%	14400.01
	3	41	471.12	100.00%	34.50	100.00%	14400.05	56.92%	100.00%	14400.58	69.15%	14400.01
	5	68	631.65	99.88%	35.31	100.00%	14400.02	23.98%	99.83%	14400.02	89.42%	14400.10
	7	95	735.20	100.00%	29.10	100.00%	14400.20	10.53%	99.85%	14400.06	96.02%	14400.01
	10	135	808.01	100.00%	23.11	100.00%	14400.04	0.57%	100.00%	14400.04	99.44%	14400.01
	13	176	812.59	100.00%	11.79	100.00%	0.46	0.00%	100.00%	186.47	100.00%	190.42
	15	203	812.59	100.00%	12.90	100.00%	0.47	0.00%	99.99%	105.22	100.00%	68.38
	20	271	812.59	100.00%	8.92	100.00%	0.47	0.00%	100.00%	21.78	100.00%	9.08
	25	338	812.59	100.00%	8.42	100.00%	0.47	0.00%	100.00%	11.50	100.00%	7.02
	30	406	812.59	100.00%	8.32	100.00%	0.47	0.00%	100.00%	5.82	100.00%	6.98
1888rte_api	1	19	310.43	99.31%	106.29	100.00%	14400.02	122.91%	96.86%	14400.03	47.88%	14400.02
	3	57	628.63	100.00%	30.63	100.00%	14400.05	22.79%	99.49%	14400.01	68.16%	14400.01
	5	94	755.83	100.00%	28.23	100.00%	14400.03	4.93%	99.47%	14400.04	93.57%	14400.01
	7	132	796.28	100.00%	23.25	100.00%	14400.06	1.04%	99.74%	14400.06	99.26%	14400.02
	10	189	804.53	100.00%	12.00	100.00%	0.75	0.00%	100.00%	641.89	100.00%	31.14
	13	245	804.53	100.00%	8.73	100.00%	0.76	0.00%	100.00%	119.60	100.00%	10.60
	15	283	804.53	100.00%	8.31	100.00%	0.74	0.00%	100.00%	83.06	100.00%	10.24
	20	378	804.53	100.00%	8.24	100.00%	0.73	0.00%	100.00%	27.09	100.00%	9.41
	25	472	804.53	100.00%	8.30	100.00%	0.74	0.00%	100.00%	25.81	100.00%	9.08
	30	566	804.53	100.00%	8.02	100.00%	0.73	0.00%	100.00%	5.54	100.00%	9.62
1951rte_api	1	20	418.14	100.00%	109.68	100.00%	14400.11	106.90%	98.00%	14400.02	54.80%	14400.02
	3	59	777.21	100.00%	42.68	100.00%	14400.04	23.11%	99.85%	14400.02	76.25%	14400.01
	5	98	935.91	100.00%	32.71	100.00%	14400.06	5.01%	99.71%	14400.33	90.71%	14400.01
	7	137	978.88	100.00%	28.69	100.00%	14400.07	1.22%	99.58%	14400.06	88.29%	14400.02
	10	195	993.09	100.00%	18.06	100.00%	17.78	0.01%	100.00%	792.13	100.00%	228.47
	13	254	993.11	100.00%	14.00	100.00%	0.76	0.00%	100.00%	341.08	100.00%	17.00
	15	293	993.11	100.00%	13.84	100.00%	3.26	0.00%	100.00%	108.54	100.00%	11.90
	20	390	993.11	100.00%	11.57	100.00%	1.65	0.00%	99.99%	13.96	100.00%	11.69
	25	488	993.11	100.00%	11.26	100.00%	0.79	0.00%	100.00%	12.06	100.00%	11.68
	30	585	993.11	100.00%	11.07	100.00%	0.83	0.00%	100.00%	5.61	100.00%	10.13

Table 3.6: Network flow restriction results on the 'sad' congested cases. The columns are labeled as in Table 3.3.

Instance	Budget (%)	# of Relays	Best Known LB	NFLB		Question 2			Question 1			
				Quality	Time (s)	Problem (E.2) Quality	Problem (E.2) Time (s)	Problem (E.2) Gap	Problem (E.2) Quality	Problem (E.2) Time to NFLB (s)	Problem (E.1) Quality	Problem (E.1) Time to NFLB (s)
1354pegase_sad	1	14	237.57	97.60%	230.54	97.60%	14400.03	136.01%	100.00%	14400.03	41.14%	14400.02
	3	41	533.47	100.00%	47.20	100.00%	14400.06	29.97%	98.98%	14400.01	63.64%	14400.01
	5	68	653.14	100.00%	22.59	100.00%	14400.04	9.28%	99.64%	14400.36	89.63%	14400.01
	7	95	698.13	100.00%	21.08	100.00%	14400.06	6.21%	99.88%	14400.03	98.69%	14400.01
	10	135	735.70	100.00%	21.86	100.00%	14400.03	0.78%	100.00%	14400.03	99.94%	14400.02
	13	176	741.46	100.00%	13.25	100.00%	0.46	0.00%	100.00%	173.48	100.00%	62.05
	15	203	741.46	100.00%	11.88	100.00%	0.45	0.00%	99.99%	98.99	100.00%	8.51
	20	271	741.46	100.00%	10.23	100.00%	0.46	0.00%	100.00%	20.86	100.00%	13.63
	25	338	741.46	100.00%	8.62	100.00%	0.46	0.00%	100.00%	8.16	100.00%	9.98
	30	406	741.46	100.00%	8.71	100.00%	0.45	0.00%	100.00%	3.14	100.00%	11.21
1888rte_sad	1	19	301.49	100.00%	53.02	100.00%	14400.09	69.67%	100.00%	14400.02	32.14%	14400.03
	3	57	525.74	100.00%	28.91	100.00%	14400.03	8.31%	99.94%	14400.03	84.64%	14400.02
	5	94	572.77	100.00%	24.08	100.00%	14400.20	2.61%	99.99%	14400.05	96.22%	14400.03
	7	132	591.76	100.00%	20.97	100.00%	14400.24	0.73%	99.99%	14400.04	98.53%	14400.02
	10	189	596.07	100.00%	11.95	100.00%	3.40	0.00%	100.00%	435.82	100.00%	43.82
	13	245	596.07	100.00%	8.38	100.00%	2.86	0.00%	100.00%	49.02	100.00%	11.23
	15	283	596.07	100.00%	8.65	100.00%	2.21	0.00%	100.00%	23.46	100.00%	15.20
	20	378	596.07	100.00%	8.43	100.00%	3.22	0.00%	100.00%	21.50	100.00%	10.79
	25	472	596.07	100.00%	8.06	100.00%	3.21	0.00%	100.00%	6.15	99.99%	8.61
	30	566	596.07	100.00%	8.14	100.00%	2.39	0.00%	100.00%	2.94	99.99%	9.31
1951rte_sad	1	20	451.75	100.00%	44.28	100.00%	14400.03	61.89%	99.94%	14400.03	49.50%	14400.01
	3	59	743.18	100.00%	37.95	100.00%	14400.04	9.03%	100.00%	14400.02	33.21%	14400.02
	5	98	810.79	100.00%	27.14	100.00%	14400.11	2.82%	100.00%	14400.05	85.76%	14400.02
	7	137	834.51	100.00%	30.46	100.00%	14400.03	0.86%	100.00%	14400.03	99.27%	14400.01
	10	195	844.27	100.00%	21.74	100.00%	9.57	0.00%	100.00%	812.21	100.00%	94.47
	13	254	844.27	100.00%	14.77	100.00%	0.78	0.00%	99.99%	26.83	100.00%	19.76
	15	293	844.27	100.00%	11.44	100.00%	0.77	0.00%	100.00%	25.56	100.00%	19.05
	20	390	844.27	100.00%	11.69	100.00%	0.77	0.00%	100.00%	10.62	100.00%	12.25
	25	488	844.27	100.00%	11.36	100.00%	0.77	0.00%	100.00%	9.69	100.00%	10.35
	30	585	844.27	100.00%	11.08	100.00%	0.77	0.00%	100.00%	10.94	100.00%	10.45

3.5.3 Quality of NFLB

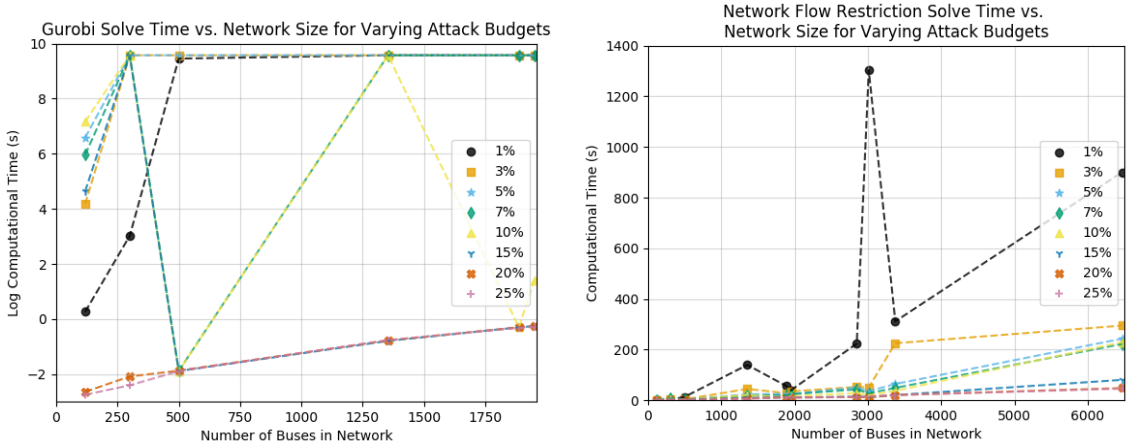
Without a nontrivial upper bound on the worst-case relay attack problem, we cannot comment precisely on the quality of the network flow restriction. However, in comparisons with the lower bound attained from solving with a heuristic bound on the dual variables, we see that in 113 out of 120 instances, NFLB was the best bound. In the 7 instances where NFLB was not the best lower bound, it was 89.25%, 91.73%, 99.77%, 99.06%, 99.88%, 99.31%, and 97.60% of the best load shed found. The budgets for which there is a gap between the best-known solution and the network flow restriction solution tend to be small. This is consistent with the bound from Theorem 1 since for these budgets there is relatively little load shed, meaning that the ℓ_1- norm of the injections is likely quite large relative to its maximum possible value for the instance (when all the load is served), making the right-hand side of (3.12) large. In this case, the theory suggests that network flow is not as good of an approximation of DCOPF. However, at least for smaller network sizes, Gurobi is able to solve (E.2) for smaller attack budgets with a heuristic bound on the dual variables, and might be a better option. For larger network sizes, even though we sometimes see a slight gap between the NFLB and (E.1) or (E.2), the network flow solution still appears to be of extremely good quality. Additionally, for these networks, (E.1) and (E.2) do not scale well enough to be computationally tractable: Among the 90 larger instances tested, (E.2) fails to achieve the NFLB within 4 hours in 25 instances, and (E.1) fails to do so in 39 instances. Last, note that even in the congested variations of the test networks shown in Tables 3.5 and 3.6, the quality of NFLB is good despite the theoretical results not holding.

3.5.4 Computational Tractability of Algorithm 1

In Tables 3.3, 3.4, 3.5, and 3.6, Algorithm 1 takes less than 4 minutes on all of the instances of the problem tested. In Table 3.7, Algorithm 1 takes less than 22 minutes in all cases, and often takes less than 5 minutes. In contrast, when solving (E.2), Gurobi times out without proving optimality within 4 hours for the hardest budgets on all but the smallest

Table 3.7: Network flow restriction results on large test cases. For each instance, we show results for 10 different budgets for the percentage of relays that can be attacked. The load shed attained from the solution given by Algorithm 1 is in the "NFLB" column and the computational time to get NFLB is shown in the last column.

Instance	Budget (%)	# of Relays	NFLB	NFLB Time (s)
2848rte	1	28	282.87	223.68
	3	85	470.57	52.62
	5	142	510.74	50.72
	7	199	530.51	42.94
	10	285	538.36	28.64
	13	370	538.35	22.20
	15	427	538.39	14.30
	20	570	538.39	13.08
	25	712	538.39	12.50
	30	854	538.39	17.30
3012wp	1	30	93.74	1302.35
	3	90	205.68	51.01
	5	151	254.73	35.87
	7	211	267.99	27.33
	10	301	271.73	19.61
	13	392	271.96	13.51
	15	452	271.96	12.11
	20	602	271.96	11.87
	25	753	271.96	15.85
	30	904	271.96	15.63
3375wp	1	34	203.44	311.63
	3	101	372.36	225.19
	5	169	470.08	64.40
	7	236	505.73	49.03
	10	337	519.36	37.04
	13	439	520.99	31.44
	15	506	520.99	20.61
	20	675	520.99	20.68
	25	844	520.99	20.29
	30	1012	520.99	19.68
6468rte	1	65	554.61	899.85
	3	194	825.38	294.70
	5	323	889.38	243.10
	7	453	924.60	222.20
	10	647	948.15	229.43
	13	841	951.54	86.39
	15	970	951.54	80.63
	20	1294	951.57	48.33
	25	1617	951.57	45.94
	30	1940	951.57	44.39



(a) Computational times for solving (E.2) with heuristic M.

(b) Computational times for the network flow restriction.

Figure 3.3: Computational times for various budgets for both Gurobi with the dual bound set to 2 and for the network flow restriction. The first plot shows the data from the "Question 1: Problem (E.2) Time" column of Tables 3.3 and 3.4, and the second plot shows the data from the "NFLB Time" column from Tables 3.3, 3.4, and 3.7.

test case. For the nine larger cases, Gurobi takes more than 4 hours to find a solution of the same quality as NFLB using problem (E.1) and with the heuristic value of M in problem (E.2). In essence, we see that scaling up the size of the network for difficult attack budgets is not feasible solving a linearization of the single-level reformulation of (3.1), even with small heuristic bounds on the duals. However, we can easily find what we believe to be a good-quality solution for even a 6,468 bus network using Algorithm 1.

These observations are visualized in Figure 3.3: In Figure 3.3a we plot on a log scale the computational times to solve (E.2) linearized using the heuristic value of M. There is noise in the 7% and 10% budgets because the most difficult budgets in that range depend on the particular network, not just the number of nodes. However, in general we see that, even for the easier very large budgets, the solve times appear to scale exponentially. For the smaller, more difficult budgets, we hit the 4-hour time limit for most of the networks. In Figure 3.3b, we plot computational times for all ten of our test networks on a linear scale. The scaling for this method appears to be roughly linear in the size of the network, where the lower budgets are more difficult and the higher budgets tend to be easier. The spike for

the 1% budget on the 3,012 bus instance is consistent for different seeds: It appears to be an anomaly in terms of difficulty for Gurobi.

Overall, we find NFLB to be approximately 150 times faster than using Gurobi. We arrive at this number by taking the average of the ratio of the time for Gurobi to reach NFLB using Problem (E.2) and the time to compute NFLB over the 120 instances tested. In summary, we see through our computational experiments that the most difficult instances of the worst-case relay attack problem are for mid-range budgets on large networks. Solving the traditional linearized single-level formulation including DCOPF in the inner problem does not scale well, even when the bound on the duals is as small as 2 or 3. In contrast, we are able to solve challenging budgets on networks up to 6,468 nodes in less than 25 minutes using the network flow restriction. To the extent it is ascertainable, the quality of solutions is good.

3.6 Setting $M = 1$ in Problem (E.2)

While the previous section showed NFLB is much faster than solving either (E.2) or (E.1), there remains a question as to whether it is only the value of M used in (E.2) that hurts Gurobi's performance or if the relaxation of DCOPF to capacitated network flow also contributes. To test this, we also run our experiments for Question 1 on problem (E.2) setting $M = 1$. This means that Problem (E.2) will have a tighter continuous relaxation, though this may cut off the solution from the NFLB.

The results of this experiment are shown in Tables 3.8-3.10. For comparison, we reprint the results from Tables 3.3-3.6 for the NFLB as well as for Question 1 and Problem (E.2) in these tables. We report the results for solving (E.2) with $M = 1$ in the last two columns of each of the tables. Note that, due to solver optimality tolerances, it is possible to get a new best-known solution from this experiment. This occurred in three instances, for the 1951rte case with a budget of 10% of the relays, and for the 1951rte_api case with budgets of 3% and 10% of the relays. The new best-known bounds for these instances are 777.87, 993.11,

Table 3.8: Comparison of solving Problem (E.2) with $M = 1$ to the results from Section 3.5.2. The first six columns reprint results from Tables 3.3 and 3.4, and the last two give results addressing Question 1, but setting $M = 1$.

Instance	Budget (%)	**NFLB**		**Question 1**		**Question 1,** $M = 1$	
		Quality	Time (s)	Problem (E.2) Quality	Problem (E.2) Time to NFLB (s)	Problem (E.2) Quality	Problem (E.2) Time to NFLB (s)
118	1	89.25%	1.02	100.00%	0.32	89.45%	0.14
	3	100.00%	1.14	100.00%	3.70	100.00%	1.42
	5	100.00%	1.42	100.00%	28.83	100.00%	3.75
	7	100.00%	1.12	100.00%	10.24	100.00%	1.66
	10	100.00%	1.25	100.00%	142.95	100.00%	4.00
	13	100.00%	1.71	100.00%	8.04	100.00%	2.31
	15	100.00%	1.16	100.00%	3.27	100.00%	4.06
	20	100.00%	0.80	100.00%	0.33	100.00%	0.06
	25	100.00%	0.84	100.00%	0.29	100.00%	0.07
300	1	91.73%	2.26	100.00%	5.35	92.40%	1.29
	3	99.77%	3.71	100.00%	99.53	99.77%	4.04
	5	100.00%	6.47	100.00%	385.57	100.00%	12.03
	7	100.00%	8.84	100.00%	538.82	100.00%	255.65
	10	100.00%	5.90	100.00%	199.14	100.00%	23.06
	13	100.00%	3.62	100.00%	355.72	100.00%	53.22
	15	100.00%	3.18	100.00%	212.62	100.00%	8.16
	20	100.00%	2.35	100.00%	19.44	100.00%	2.02
	25	100.00%	2.07	100.00%	2.32	100.00%	0.20
500tamu	1	100.00%	13.71	100.00%	3872.70	100.00%	339.26
	3	100.00%	3.92	100.00%	44.53	100.00%	5.79
	5	100.00%	2.88	100.00%	44.11	100.00%	10.39
	7	100.00%	2.87	100.00%	14.38	100.00%	3.03
	10	100.00%	2.17	100.00%	2.82	100.00%	0.26
	13	100.00%	2.25	100.00%	2.16	100.00%	0.82
	15	100.00%	2.89	100.00%	1.02	100.00%	0.31
	20	100.00%	2.08	100.00%	3.09	100.00%	0.69
	25	100.00%	2.15	100.00%	1.66	100.00%	0.38
1354pegase	1	100.00%	139.59	97.74%	14400.02	99.84%	14400.03
	3	100.00%	44.77	98.91%	14400.05	99.65%	14400.03
	5	100.00%	23.55	99.18%	14400.06	100.00%	14400.04
	7	100.00%	18.53	99.87%	14400.26	100.00%	14400.04
	10	100.00%	20.94	100.00%	14400.03	100.00%	14400.23
	13	100.00%	12.55	100.00%	179.52	100.00%	26.82
	15	100.00%	9.30	100.00%	85.14	100.00%	20.92
	20	100.00%	7.03	99.99%	15.57	100.00%	10.74
	25	100.00%	6.23	100.00%	9.65	100.00%	4.49

Table 3.9: Comparison of solving Problem (E.2) with $M = 1$ to the results from Section 3.5.2. The first six columns reprint results from Tables 3.4 and 3.5, and the last two give results addressing Question 1, but setting $M = 1$.

Instance	Budget (%)	**NFLB**		**Question 1**		**Question 1,** $M = 1$	
		Quality	Time (s)	Problem (E.2) Quality	Problem (E.2) Time to NFLB (s)	Problem (E.2) Quality	Problem (E.2) Time to NFLB (s)
1888rte	1	100.00%	45.70	100.00%	8629.13	100.00%	1262.56
	3	100.00%	28.92	100.00%	3790.54	100.00%	388.33
	5	100.00%	27.58	99.97%	14400.18	100.00%	280.47
	7	100.00%	21.69	99.99%	14400.07	100.00%	226.05
	10	100.00%	12.60	100.00%	360.94	100.00%	19.30
	13	100.00%	9.42	100.00%	43.09	100.00%	6.65
	15	100.00%	8.83	100.00%	15.65	100.00%	4.37
	20	100.00%	8.38	100.00%	5.70	100.00%	3.55
	25	100.00%	8.12	100.00%	14.02	100.00%	3.74
1951rte	1	99.06%	42.70	100.00%	14400.03	99.47%	14400.07
	3	100.00%	35.71	99.89%	14400.04	100.00%	14400.21
	5	100.00%	26.35	100.00%	14400.04	100.00%	14400.04
	7	100.00%	26.13	100.00%	14400.06	100.00%	14400.05
	10	100.00%	18.37	100.00%	972.18	100.00%	62.22
	13	100.00%	13.43	100.00%	47.39	100.00%	14.31
	15	100.00%	11.70	100.00%	44.41	99.99%	11.63
	20	100.00%	11.21	100.00%	6.63	100.00%	6.23
	25	100.00%	11.19	100.00%	10.19	100.00%	3.30
1354pegase_api	1	100.00%	60.56	100.00%	14400.04	99.33%	14400.04
	3	100.00%	34.50	100.00%	14400.58	99.91%	14400.27
	5	99.88%	35.31	99.83%	14400.02	99.93%	14400.12
	7	100.00%	29.10	99.85%	14400.06	100.00%	14400.06
	10	100.00%	23.11	100.00%	14400.04	100.00%	14400.04
	13	100.00%	11.79	100.00%	186.47	100.00%	40.33
	15	100.00%	12.90	99.99%	105.22	100.00%	13.32
	20	100.00%	8.92	100.00%	21.78	100.00%	8.63
	25	100.00%	8.42	100.00%	11.50	100.00%	5.90
1888rte_api	1	99.31%	106.29	96.86%	14400.03	99.18%	14400.03
	3	100.00%	30.63	99.49%	14400.01	100.00%	14400.04
	5	100.00%	28.23	99.47%	14400.04	100.00%	14400.04
	7	100.00%	23.25	99.74%	14400.06	100.00%	14400.05
	10	100.00%	12.00	100.00%	641.89	100.00%	25.40
	13	100.00%	8.73	100.00%	119.60	100.00%	14.73
	15	100.00%	8.31	100.00%	83.06	100.00%	5.39
	20	100.00%	8.24	100.00%	27.09	100.00%	5.31
	25	100.00%	8.30	100.00%	25.81	100.00%	3.64

Table 3.10: Comparison of solving Problem (E.2) with $M = 1$ to the results from Section 3.5.2. The first six columns reprint results from Tables 3.5 and 3.6, and the last two give results addressing Question 1, but setting $M = 1$.

Instance	Budget (%)	**NFLB**		**Question 1**		**Question 1,** $M = 1$	
		Quality	Time (s)	Problem (E.2) Quality	Problem (E.2) Time to NFLB (s)	Problem (E.2) Quality	Problem (E.2) Time to NFLB (s)
1951rte_api	1	100.00%	109.68	98.00%	14400.02	99.21%	14400.04
	3	99.92%	42.68	99.77%	14400.02	100.00%	14400.03
	5	100.00%	32.71	99.71%	14400.33	100.00%	14400.06
	7	100.00%	28.69	99.58%	14400.06	100.00%	14400.04
	10	99.99%	18.06	100.00%	792.13	100.00%	80.16
	13	100.00%	14.00	100.00%	341.08	100.00%	20.98
	15	100.00%	13.84	100.00%	108.54	100.00%	8.98
	20	100.00%	11.57	99.99%	13.96	100.00%	5.28
	25	100.00%	11.26	100.00%	12.06	100.00%	4.63
1354pegase_sad	1	97.60%	230.54	100.00%	14400.03	97.54%	14400.03
	3	100.00%	47.20	98.98%	14400.01	100.00%	14400.03
	5	100.00%	22.59	99.64%	14400.36	100.00%	14400.03
	7	100.00%	21.08	99.88%	14400.03	100.00%	14400.43
	10	100.00%	21.86	100.00%	14400.03	100.00%	14400.04
	13	100.00%	13.25	100.00%	173.48	100.00%	23.64
	15	100.00%	11.88	99.99%	98.99	100.00%	21.59
	20	100.00%	10.23	100.00%	20.86	100.00%	5.64
	25	100.00%	8.62	100.00%	8.16	100.00%	3.63
1888rte_sad	1	100.00%	53.02	100.00%	14400.02	99.42%	14400.03
	3	100.00%	28.91	99.94%	14400.03	100.00%	14400.03
	5	100.00%	24.08	99.99%	14400.05	100.00%	14400.04
	7	100.00%	20.97	99.99%	14400.04	100.00%	14400.07
	10	100.00%	11.95	100.00%	435.82	100.00%	26.05
	13	100.00%	8.38	100.00%	49.02	100.00%	9.05
	15	100.00%	8.65	100.00%	23.46	100.00%	4.79
	20	100.00%	8.43	100.00%	21.50	100.00%	4.20
	25	100.00%	8.06	100.00%	6.15	100.00%	6.08
1951rte_sad	1	100.00%	44.28	99.94%	14400.03	99.18%	14400.03
	3	100.00%	37.95	100.00%	14400.02	100.00%	14400.11
	5	100.00%	27.14	100.00%	14400.05	100.00%	14400.06
	7	100.00%	30.46	100.00%	14400.03	100.00%	14400.05
	10	100.00%	21.74	100.00%	812.21	99.99%	34.87
	13	100.00%	14.77	99.99%	26.83	100.00%	17.35
	15	100.00%	11.44	100.00%	25.56	100.00%	5.97
	20	100.00%	11.69	100.00%	10.62	100.00%	6.99
	25	100.00%	11.36	100.00%	9.69	100.00%	3.13

and 844.25 respectively. We calculate solution quality with respect to these numbers in Tables 3.9 and 3.10.

Overall, setting $M = 1$ in Problem (E.2) is consistently much faster to solve than the values of 2 and 3 from Section 3.5.2. However, except for in the 118-bus case and in easier budgets on the other cases, calculating NFLB is still much faster. We can therefore conclude that bounding the defender problem duals by 1 is not the only advantage of the network flow restriction. While tighter values of M do lead to faster solves, dropping the constraints corresponding to Ohm's law appears to also have computational advantages.

3.7 Conclusion

In this chapter, we analyzed a restriction of the worst-case relay attack problem which has theoretical guarantees on uncongested networks and which we have also shown empirically to provide a high-quality lower bound, even on congested networks. We have shown that, in addition to the apparent tightness of the lower bound, the network flow restriction can be solved efficiently and to scale with a commercial MIP solver. We suspect this is due in part to the fact that the network flow restriction can be linearized with big-M values of 1, and also in part to the familiar, well-studied structure of network flow itself.

In future work, there is a need to consider upper bounds for this problem and to improve the scalability of exact methods. Additionally, higher-complexity restoration models have been shown to be important for $N - k$ models when k is large (Coffrin, Bent, et al., 2019), so there is a need to find scalable solution methods when the defender problem includes elements such as nonlinear approximations of the AC power flow equations, bus shunts, and line charging. Also, the network flow approximation for DCOPF could be used in place of DCOPF in numerous other problems for both power systems operations and security. Since power systems are rarely congested in practice, it is likely that this approximation can be of use in order to scale up other problems which currently rely on DCOPF. Last, an interesting open question is if it is possible to modify the thermal limits, generation

bounds, or demands in the network flow problem so that it would be a better approximation of DCOPF, even in congested networks.

Acknowledgments

The work in this chapter was in collaboration with Santanu Dey. We would also like to thank Bryan Arguello, Anya Castillo, Jared Gearhart, and Cynthia Phillips for helpful discussions during this work.

This work was supported by Sandia National Laboratories' Laboratory Directed Research and Development (LDRD) program. Sandia National Laboratories is a multimission laboratory managed and operated by National Technology and Engineering Solutions of Sandia, LLC, a wholly owned subsidiary of Honeywell International, Inc., for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-NA0003525. SAND NO. 2021-10211 O. The views expressed in the article do not necessarily represent the views of the U.S. Department of Energy or the United States Government.

CHAPTER 4
A COVERING DECOMPOSITION ALGORITHM FOR POWER GRID CYBER-NETWORK SEGMENTATION

We present a trilevel interdiction model for optimally segmenting the Supervisory Control and Data Acquisition (SCADA) network controlling an electric power grid. In this formulation, we decide how to partition nodes of the SCADA network in order to minimize the shedding of load from a worst-case cyberattack, assuming that the grid operator has the opportunity for recourse mitigating attack damage. The model is unique in that it couples the cyber and physical networks, using the physical operation of the grid to inform the cyber network segmentation decisions. This feature leads to two violations of assumptions made in much of the prior literature on trilevel interdiction: first, the network designer (first player) does not have the ability to make any components of the SCADA or physical networks invulnerable to the second player's attack. Instead, the designer makes some attacks more expensive for the attacker. Second, it is possible for the network designer to make certain attacker solutions infeasible because he can make them exceed the attacker's budget. In this chapter, we present a solution procedure for our formulation that is an adaptation of a covering decomposition algorithm for bilevel interdiction. We show through an empirical study on grids with up to 2,000 buses that this is the first method capable of solving the network segmentation model on realistically sized power grids.

4.1 Introduction

This chapter describes a trilevel optimization approach to planning the defense of the computer networks controlling electrical power grids. Electrical power grids, like all modern energy distribution infrastructures, are controlled and monitored by SCADA systems. These systems are candidates for cyberattacks with potentially severe physical con-

sequences (Genge et al., 2017). Incidents such as the Spring 2021 Colonial Pipeline disruption and the 2015 Ukraine power system cyberattack underscore the likelihood and potential impact of hackers compromising SCADA systems. As the importance of electric power increases in proportion to other forms of energy (e.g., electric vehicles), "smart grid" capabilities expand, and power grid control becomes increasingly networked, electric power grids are becoming increasingly vulnerable and important potential targets. This chapter describes a modeling approach and accompanying solution algorithm for mitigating the impact of attacks on power grid control infrastructures.

Our approach does not assume that any part of the control network or power grid infrastructure can be absolutely hardened against assailants. Instead, the goal is to increase the difficulty of an attacker being able to have a significant impact on power delivery. In the model developed in this chapter, the basic tool to increase the difficulty of attack is partitioning nodes of the control network into multiple "enclaves," a process called *network segmentation*. Communication among different enclaves is limited, and to control or disable any particular relay in the physical power distribution network, we assume that an attacker must compromise all the control network enclaves to which it is assigned. Each additional enclave that an assailant must compromise to execute an attack is presumed to require time and effort on their part, and also to raise the probability of detection, all of which may be interpreted as costs to the attacker. The general goal of the defender is to configure the enclaves so that an attacker must penetrate a relatively large number of them in order to have a significant effect on the amount of power delivered by the physical network, thus making it harder to execute a high-impact attack. Using network segmentation to mitigate attacks is recommended in U.S. and Australian government documents such as US Department of Homeland Security (2016), Pillitteri and Brewer (2014), Stouffer et al. (2016), and Australian Cyber Security Centre (2020).

This chapter presents a model and accompanying solution method for deciding where to best place enclaves in a power grid's SCADA network. The model uses a three-level Stackelberg game framework whose moves proceed as follows:

1. Subject to various budget constraints, the first player, whom we refer to as the *designer*, divides components of the SCADA network into enclaves.

2. The second player, whom we call the *attacker*, then decides which enclaves to compromise and disables the physical elements of the power network controlled by those enclaves. The total number of enclaves compromised is subject to a budget constraint.

3. Finally, by reconfiguring the uncompromised portions of the power network, the third player, whom we call the *defender*, attempts to fulfill as much power demand as is still possible. This process is called "re-dispatch".

The amount of consumer power demand left unmet after the last move is referred to as "load shed". The designer and defender objectives are to minimize load shed and the attacker's objective is to maximize it. Therefore, we are considering an *interdiction* problem, that is, a multilevel optimization problem in which the levels alternate between exactly opposing objectives. Such problems are common in operations research, but in our case the situation is especially challenging because there are three decision levels instead of the more customary two.

Trilevel interdiction formulations of infrastructure security problems have been investigated before, as discussed below in Section 4.1.1. The model presented here builds on Arguello et al. (2021), where the designer decisions model realistic cybersecurity measures, focusing on specifying the structure of the SCADA network through segmentation. This approach differs from other trilevel formulations where the designer selects particular physical network assets to completely "harden" or make invulnerable to compromise by the attacker. While the trilevel interdiction problem we solve is in essence the same as that

proposed by Arguello et al. (2021), we use a different solution procedure that is able to solve larger and more difficult problem instances. To make this procedure more workable, we use a modified formulation with a different attacker strategy representation.

Trilevel models such as we consider here lead to particularly challenging forms of optimization problems. For example, even solving linear bilevel problems is NP-hard (Jeroslow, 1985). Here, we take the standard approach of conceptualizing a trilevel problem as a bilevel optimization problem whose second stage is itself a bilevel problem, and then applying various bilevel optimization techniques. Fundamentally, there are three basic approaches to bilevel optimization:

Follower optimality constraints: This approach involves formulating a single-level optimization model that enforces optimality of the follower-controlled variables through explicit constraints. One then formulates the entire bilevel problem as optimization of the leader objective over a feasible region that includes these constraints. For a convex follower problem, the typical approach involves enforcing the Karush-Kuhn-Tucker (KKT) conditions by explicitly including both primal and dual variables for the follower and requiring complementary slackness through disjunctions or auxiliary integer variables combined with "big-M" constraints. In the case of interdiction problems, however, one may be able to take a simpler route in which the follower problem is essentially replaced with its dual, which is solved in conjunction with the leader problem; this approach is known as *dualize and combine*.

Follower optimality by generating cutting planes: This approach starts with the *high-point relaxation*, in which the follower objective is effectively eliminated and the leader has full control of the follower variables; both the leader or follower variables may be subject to integrality constraints. While solving the resulting mixed-integer program by branch and cut,

one incrementally introduces cutting planes that invalidate solutions that are suboptimal for the follower. This technique is the core of the approach taken in the software of Fischetti et al. (2017) and Tahernejad et al. (2020).

Response-function methods: These techniques, closely related to Benders decomposition, build a progressively more accurate functional model of the follower response to various leader decisions. Each time the model is improved, one computes an optimal leader decision with respect to the current model, finds an optimal follower response, and uses that response to further improve the follower model. Wood (2011) presents such a technique for bilevel interdiction models with convex inner problems. Tang et al. (2016) consider interdiction models with integrality requirements in the inner problem. Lozano and Smith (2017b) propose a similar algorithm for general bilevel problems.

Yue et al. (2019) combine elements of two of the above approaches into a column-and-constraint generation algorithm. As in the follower-optimality-by-cutting-planes approach, the algorithm iteratively refines the high-point relaxation, but — similarly to response-function methods — this refinement is effected by incrementally generating follower solutions. This technique can solve problems in which the leader can make the follower problem infeasible, unlike the branch-and-cut techniques mentioned above. We also refer the interested reader to Smith and Song (2020) for a survey of algorithms for network interdiction problems.

In our trilevel model, the third stage is a continuous, convex power flow problem for which we use a network flow relaxation, an approximation that we justify through the results in Chapter 3; see Section 4.3.2 for a more detailed discussion. The resulting convexity of the last stage makes it natural to use an approach in the follower-optimality-constraint class to link the second and third stages. Since we have an interdiction model, we are able

to use the dualize-and-combine technique to merge the second and third stages, essentially replacing the re-dispatch problem with its dual, solved in conjunction with the attack decisions. We call the resulting model the "attacker MIP" (since the attacker's decisions are discrete).

Using this kind of merger of the second and third stages, Arguello et al. (2021) attempt to solve the full trilevel problem by applying the follower-optimality-by-cutting-planes method, as embodied in the software of Fischetti et al. (2017), to the mixed-integer bilevel problem whose first stage consists of the network segmentation decisions and whose second stage is the attacker MIP. However, this approach has difficulty solving instances of realistic size, likely because the high-point relaxation used by the cutting plane approach is particularly weak for interdiction problems: since the leader and follower have opposite objectives, the high-point relaxation chooses solutions that are very far from optimal for the follower, and large numbers of cutting planes must be incrementally added to guide the solution toward follower optimality. These difficulties agree with the observations regarding the unsuitability of general bilevel methods for interdiction problems made in Wood (2011) and Hua et al. (2019); the latter state that "relaxations based on the high-point problem are weak for an interdiction problem with its directly conflicting objective functions."

Here, we similarly use dualize and combine to link the second and third stages, but coordinate the first and second stages differently, using a specialized response-function technique. The result may be viewed as a trilevel adaptation of the covering decomposition method proposed for bilevel problems by Israeli and Wood (2002). Our algorithm maintains a set of anticipated attacks K that might be executed by the assailant, and uses the following basic loop:

1. Attempt to compute a defender configuration that "blocks" all the attacks in K, that is, makes them impossible to execute within the attacker budget. This step involves solving a feasibility problem, which in our case involves only binary variables.

2. Compute the highest-impact attack k possible on the defender configuration just computed in step 1.

3. Update the set of anticipated attacks K to include k, and return to step 1.

If there is no solution to the feasibility problem in step 1, then the algorithm terminates with the optimal defense configuration being the the best one generated so far. The intention of the method is that termination may be achieved with K containing only a tiny fraction of all possible attacks.

Our computational experiments (see Section 4.4) demonstrate that our method can find optimal solutions to problems on realistically sized power grids with up to 2,000 buses, so long as the attacker and defender budgets are relatively small (a "bus" is essentially equivalent to a node in the power network). For smaller grids, we can find optimal solutions for larger budgets.

The computational results below, which contain a grid of designer and attacker budget combinations, also suggest how the network designer might assess the impact of various designer budgets on network security. Such an assessment could then eventually lead to the selection of a particular budget and a concrete network segmentation plan.

While our approach effectively solves the same class of problem as in Arguello et al. (2021), using a response-function methodology to link the first and second stages necessitated a new, different formulation. Specifically, we do not describe potential attacks by which enclaves they penetrate, as in Arguello et al. (2021). Instead, each attack is effectively a list of the physical power network elements the attacker needs to control, from which the required set of enclave penetrations can be deduced for any particular control network configuration. This change obviates numerous potential model symmetries and makes a response-function approach practicable.

In summary, this chapter makes three closely interrelated contributions to the defense of power-grid SCADA networks: (1) reformulation of the general approach proposed in Arguello et al. (2021) to make a response-function solution method more practical, (2)

application of a response-function method, specifically an adaptation of the covering decomposition algorithm of Israeli and Wood (2002) to trilevel problems in power-grid security, and (3) the first demonstrated solution of trilevel control network segmentation models derived from realistically sized power grids. Since network segmentation formulations model power grid cybersecurity much more realistically than previous physical-component-hardening models, being able to solve network segmentation problems for realistically sized power grids is a significant advance.

4.1.1 Literature Review

There has been much interest in trilevel interdiction in the past twenty years, particularly with applications in power systems resilience (Oster et al., 2020). However, the field appears to contain relatively few truly distinct algorithmic approaches. In general, these ideas can be sorted into two broad categories: for trilevel interdiction problems where the second player's feasible region does not depend on the first player's decision, methodology from two-stage robust optimization applies. For problems where this independence property does not hold, several techniques originally developed for bilevel problems apply, most often the covering decomposition method from Israeli and Wood (2002) and the implicit enumeration approach from Scaparra and Church (2008). For problems where the third-player problem is convex, it is common to reformulate the trilevel problem as a bilevel problem by using dualize and combine on the inner two levels, after which bilevel solution algorithms can be applied directly. Otherwise, bilevel decomposition algorithms can be applied in a nested fashion. Solving the subproblem then also requires an algorithm for bilevel optimization. In the remainder of this section, we survey some trilevel interdiction problems in a variety of domains, with a focus on the methodology used to solve them. When referring to upper and lower bounds on the solution value, we will use the convention that all problems are in a min-max-min format, as in our formulation. For problems originally

having a max-min-max format, we thus reverse the meaning of "upper" and "lower" from the original publication.

Numerous papers formulate trilevel interdiction problems so that the feasible region of the second player does not depend on the first player's decision. The resulting formulations have the same basic structure as two-stage robust optimization problems, allowing the application of algorithms from the robust optimization literature. In this context, many authors have had success applying the column-and-constraint-generation algorithm from Zeng and Zhao (2013). For example, Yuan et al. (2014) apply this algorithm to a defender-attacker-defender problem to decide which lines in a transmission grid to harden against attack. Ding et al. (2018) extend this formulation to handle multiple uncertainty sets for the attacker and again apply the Zeng and Zhao column-and-constraint-generation algorithm. Fang and Zio (2017) use the same algorithm to solve an infrastructure resilience defender-attacker-defender model, and Wu and Conejo (2017) use it to solve an electric grid defense planning problem. Lai et al. (2019) use the same approach to solve a trilevel formulation for allocating resources to defend against a coordinated cyber and physical attack on the power grid. None of these methods are directly applicable to our network segmentation formulation since they cannot handle feasibility dependence between the first two players.

Alderson et al. (2011) develop an algorithm for defender-attacker-defender problems which may also be viewed as a column-and-constraint-generation algorithm. They assume that the defender can make particular physical-grid assets impenetrable to the attacker. Doing so does not make the attacker problem infeasible, but instead means that the attacker gets no benefit from targeting impenetrable assets. This means that, again, the second-player feasible region does not depend on the first-player decision. Under these assumptions, Alderson et al. then present a similar algorithm to Zeng and Zhao (2013), generating attacks to add to a master problem that provides a lower bound. The damage from the attacks gives an upper bound. This approach is very similar to the algorithm presented here, but it is not directly applicable since our model's first player can make particular

attacks infeasible, a property that adds disjunctions to the lower-bounding problem. Furthermore, we do not assume that the first player can make physical assets invulnerable. Ouyang (2017), Ouyang and Fang (2017), Ouyang, Xu, et al. (2017), and Ouyang, Tao, et al. (2018) all adapt the algorithm of Alderson et al. (2011) to various trilevel problems to guide investments to enhance resilience of critical infrastructure.

For trilevel interdiction problems with a convex third level, it is common to use a dualize-and-combine reformulation to transform the inner bilevel problem into a single-level problem. For example, Alguacil et al. (2014) consider a power-grid hardening problem taking into account a set of plausible contingencies. Similarly to Alderson et al. (2011), their model assumes that the defender can make some network assets impenetrable. The third-level (defender) problem is a DC Optimal Power Flow (DCOPF) problem, which is convex, so they may use dualize and combine to create a bilevel problem to which they directly apply the implicit enumeration algorithm of Scaparra and Church (2008). While a variation on this method would be possible in principle for the network segmentation problem, the decision space to enumerate would be much larger because one cannot directly fortify physical network components.

Our paper is not the first to adapt the covering decomposition algorithm from Israeli and Wood (2002) to a trilevel setting. Yao et al. (2007) consider a trilevel problem to defend a power network against terrorist attacks. They apply the covering decomposition algorithm to the trilevel problem in a nested fashion: they treat the first-player problem as a feasibility problem, and again use the covering decomposition algorithm in order to find the worst-case attack solutions to the inner bilevel problem. The covering decomposition algorithm is also applied to a protection-interdiction-restoration model for grid resilience in Ghorbani-Renani, González, Barker, and Morshedlou (2020), but this methodology again depends on being able to directly protect individual physical assets. Similarly to Yao et al. (2007), they use covering decomposition in a nested fashion, although their first-player problem has an objective function.

Ghorbani-Renani, González, and Barker (2021) present a general algorithm for trilevel optimization, which they decompose into two bilevel problems: a master problem in which the second player solution is fixed, giving a lower bound, and a subproblem where the first-player solution is fixed, giving an upper bound. These bounds are closed by alternating between the two problems and adding cuts to the subproblem requiring that it does not encounter the same solution twice. This kind of approach is again only valid if the first player cannot make second-player solutions infeasible.

Lozano and Smith (2017a) present a novel backward-sampling algorithm for trilevel interdiction problems in which the sets of assets that can be protected and attacked are the same, and the previously mentioned assumption about impenetrable assets holds. This assumption allows the feasible region of the second problem to be independent of the first-player problem decision. The first two problems must be pure binary, while the third problem may have any mix of integer and continuous variables. The algorithm is based on iteratively sampling from the third player's feasible solutions, with each sample providing a restriction of the trilevel problem. This restriction is used to construct both upper and lower bounds and to identify assets that the first player must defend. However, this method is not directly applicable to our network segmentation model, since we do not assume that the first player can make some assets impenetrable. In addition, because the inner two problems of our model are easily reformulated as a single-level problem, we do not need the backward sampling methodology.

4.2 Network Segmentation Model

We now present a novel formulation of the network segmentation problem first developed in Arguello et al. (2021), with several modifications. We model the SCADA system controlling the transmission grid as a forest with depth four. *Balancing authorities* are the root nodes, and their descendants (in order) are *control centers*, *substations*, and *relays*. The relays control the lines, generators, and loads on the physical grid. Throughout this

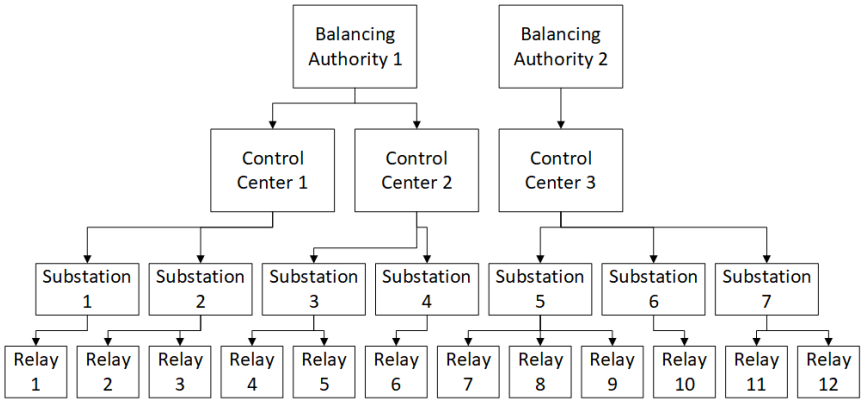


Figure 4.1: Control network structure. Balancing authorities enclaves communicate with control center enclaves which in turn communicate with substation enclaves. These then control the relays, each of which controls a set of physical grid components (i.e., lines, generators, and loads).

chapter, substations are in one-to-one correspondence with the buses on the physical grid. Figure 4.1 shows a diagram of an example control network.

The goal of the network segmentation problem is to divide selected balancing authority, control center, and substation nodes into multiple cybernetwork partitions, henceforth referred to as *enclaves*, so as to contain the load shed from a worst-case cyberattack as much as possible. Unpartitioned nodes of the original network constitute a single enclave, and the number of new enclaves that may be introduced is subject to a budget constraint.

We formulate the network segmentation problem as a trilevel optimization problem. In the first-player problem, the designer decides where to add new enclaves subject to two separate budget cardinality constraints, one for the balancing authorities and control centers combined, and one for the substations. If no enclaves are added to a node, it contains only its single, original enclave. The designer also determines parent-child relationships between the enclaves, respecting the original control hierarchy: given a node n with parent $P(n)$ in the original control network, each enclave within n must be the child of a single enclave within $P(n)$, but the designer is free to choose which one whenever $P(n)$ has multiple enclaves. The resulting, segmented control network remains a forest. For example, Figure 4.2 shows a segmented version of the "balancing authority 2" tree from Figure 4.1:

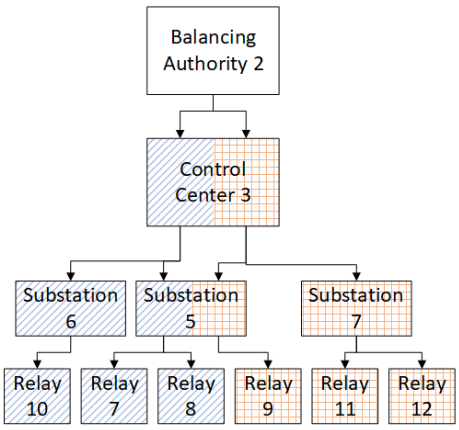


Figure 4.2: Example of a segmented portion of the control network from Figure 4.1. The balancing authority still contains one enclave, but the control center and substation 5 now contain two enclaves each. The substations and relays are patterned according to which control center enclave ultimately controls them. The enclaves in substation 5 are controlled by different parents.

here, no enclaves were added to balancing authority 2, but control center 3 and substation 5 each have one new enclave. Each of the nodes is patterned according to which control center network ultimately controls it. Note that the partition refines the original forest structure: for example, relays 7, 8, and 9 are still children of substation 5 and grandchildren of control center 3, but relay 9 is now in different control center and substation enclaves than relays 7 and 8. Figure 4.3a and Figure 4.3b show two examples of disallowed network segmentation decisions: in the former, relay 8 is the child of two different enclaves within substation 5, while in the latter, relay 7 has become a child of an enclave in substation 6 despite originally being a child of substation 7. (Figure 4.3c violates a different constraint that we will describe below).

After the designer has selected a control network segmentation scheme, the attacker maximizes load shed by compromising relays and shutting off the grid components they control. To gain access to a relay, the attacker must have access to the substation enclave that controls it. In turn, to obtain access to any enclave, he must have access to its parent. Thus, the attacker is selecting a subforest of the post-segmentation control network graph that maximizes the load shed (after the defender's response) when he controls its

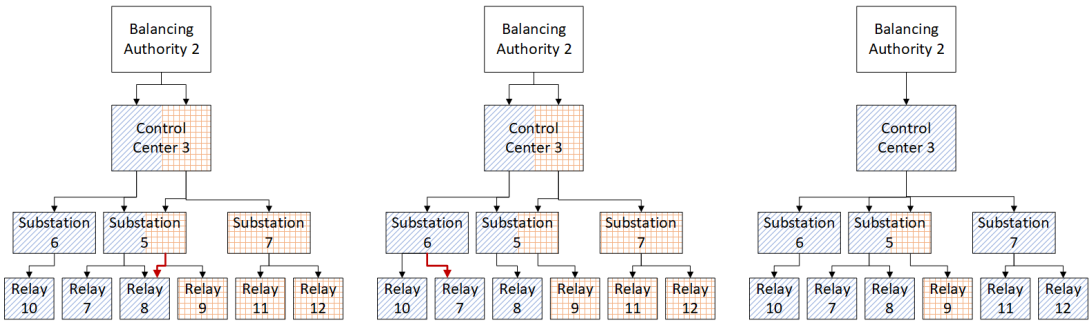


Figure 4.3: Examples of disallowed network segmentation choices. In (a), relay 8 has been assigned to both of the enclaves in substation 5, and the resulting network is no longer a forest. In (b), the original network structure is violated since relay 7 was originally a child of substation 5, but has been reassigned to substation 6. In (c), substation 5 is segmented, but both of its enclaves are assigned to the same control center enclave (see below for this restriction).

leaves. The attacker is assumed to be subject to a limit on a number of enclaves (not including relays) that he can compromise. For example, with a budget of three enclaves in the unsegmented network in Figure 4.1, the attacker might choose to compromise balancing authority 2, control center 3, and substation 5, hence gaining control of relays 7, 8, and 9. In the segmented network partially displayed in Figure 4.2, attacking the same set of relays would require a budget of five enclaves, since the attacker would need to compromise both enclaves in control center 3 and both enclaves in substation 5.

Last, in the third-player problem, the defender re-dispatches the uncompromised portions of the grid, as modeled by a capacitated network flow problem. Taken together, the entire formulation is therefore a trilevel interdiction problem with pure binary first and second actors and a convex third level. For further detail on the model, we refer the reader to Arguello et al. (2021).

We enhance the realism of the model from Arguello et al. (2021) with the additional constraint that the network designer cannot assign multiple substation enclaves within a single substation to be controlled by the same control center enclave. Figure 4.3c shows a variation of the segmentation from Figure 4.2 that violates this constraint: it assigns both

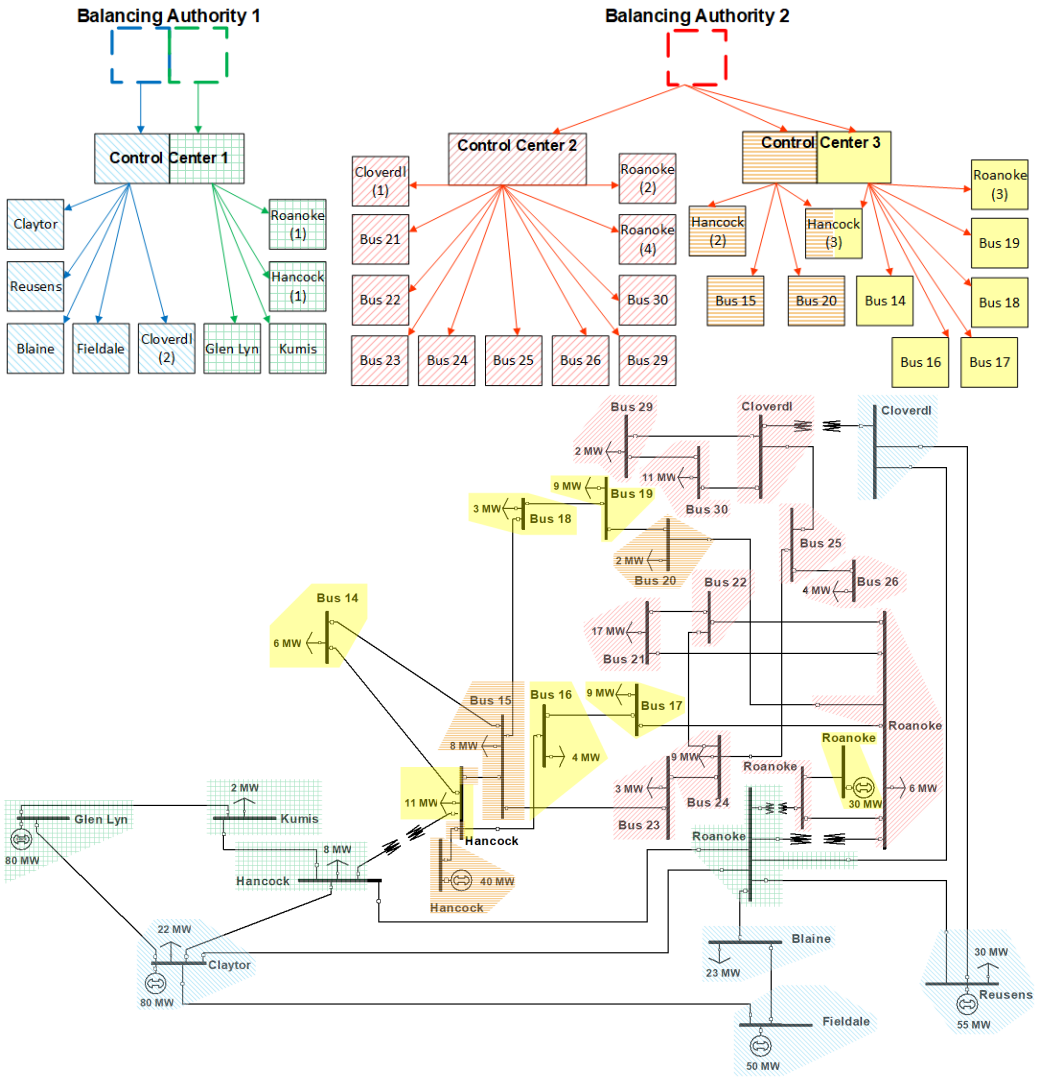


Figure 4.4: How segmentation of a 30-bus system's control network relates to the physical network. The top image displays the segmented control forest, and the lower image shows its relation to the physical grid. Each leaf node in the forest corresponds to one substation/bus on the physical grid. Substations are colored according to their parent control center enclaves.

enclaves in substation 5 to the blue (striped) enclave in control center 3. The reasoning behind this constraint is that workstations and thus network access points are located at the control center level of the SCADA network. We thus assume that an attacker will scan the network from the control center level. There is thus no point in having multiple substation enclaves controlled by the same control center enclave, since both would be fully visible to an attacker having penetrated that enclave. Note, however, that the attacker still pays a unit of budget to pivot into a substation enclave, as this action still increases his risk of detection. Last, Arguello et al. (2021) use three separate budgets in the designer problem, treating balancing authorities, control centers, and substations separately. We combine the balancing authority and control center budgets, allowing the optimization model to resolve the trade-offs between them.

In Figure 4.4, we show the control network and physical network for a segmented 30-bus system. Due to space constraints, we do not depict the relays in the communication network, so the displayed leaf nodes correspond to the substations on the physical grid, which correspond to buses in the power network.

4.2.1 Formulation

We formulate the network designer problem as if every node in the communication network has a number of potential enclaves equal to the budget at its tier of the network, plus one; specifically we use the indexing scheme that $E(n) = \{0, 1, \ldots, T\}$ for all $n \in \mathcal{B} \cup \mathcal{C}$. The designer decides which of these enclaves to "activate," that is, add to the network. We do not explicitly model the enclaves at the substation tier of the network. Instead, we assign relays to be children of control center enclaves, which determines the enclaves at the substation level because of the constraint that no control center enclave has more than one child enclave in the same substation. Our network segmentation problem is given by

$$\min_{(x,y,q,z)\in\mathcal{D}} \left\{ \max_{(\alpha,\beta,\delta,u,v,w)\in\mathcal{A}(y,q)} \left\{ \min_{(p,f,\ell)\in\mathcal{X}(u,v,w)} \left\{ \sum_{s\in\mathcal{S}} \ell_s \right\} \right\} \right\} \tag{4.1}$$

where the set $\mathcal{D}$ is defined by:

$$y_{n,i,j} \leq x_{P(n),j} \qquad \forall n \in \mathcal{C}, i \in E(n), j \in E(P(n)) \tag{4.2}$$

$$q_{r,j} \leq x_{P(P(r)),j} \qquad \forall r \in \mathcal{R}, j \in E(P(P(r))) \tag{4.3}$$

$$\sum_{j \in E(P(n))} y_{n,i,j} = x_{n,i} \qquad \forall n \in \mathcal{C}, i \in E(n) \tag{4.4}$$

$$\sum_{j \in E(P(P(r)))} q_{r,j} = 1 \qquad \forall r \in \mathcal{R} \tag{4.5}$$

$$x_{n,i} \leq \sum_{v \in C(n)} \sum_{j \in E(v)} y_{v,j,i} \qquad \forall n \in \mathcal{B}, i \in E(n) \tag{4.6}$$

$$x_{n,i} \leq \sum_{r \in R(n)} q_{r,i} \qquad \forall n \in \mathcal{C}, i \in E(n) \tag{4.7}$$

$$z_{s,i} \geq q_{r,i} \qquad \forall s \in \mathcal{S}, r \in C(s), i \in E(P(s)) \tag{4.8}$$

$$\sum_{n \in \mathcal{B} \cap \mathcal{C}} \sum_{i \in E(n) \setminus \{0\}} x_{n,i} \leq T \tag{4.9}$$

$$\sum_{s \in \mathcal{S}} \sum_{i \in E(P(s))} z_{s,i} \leq V \tag{4.10}$$

$$x_{n,i} \in \{0,1\} \qquad \forall n \in \mathcal{B} \cup \mathcal{C}, i \in E(n) \tag{4.11}$$

$$y_{n,i,j} \in \{0,1\} \qquad \forall n \in \mathcal{C}, i \in E(n), j \in E(P(n)) \tag{4.12}$$

$$q_{r,j} \in \{0,1\} \qquad \forall r \in \mathcal{R}, j \in E(P(r)) \tag{4.13}$$

Recall that $x_{n,i}$ indicates whether enclave i in node n is active and $y_{n,i,j}$ indicates whether enclave i in node n is a child of enclave j in node $P(n)$. Keeping these definitions in mind, the constraints (4.2) enforce that enclave j in the parent balancing authority $P(n)$ of some control center n must be active if it has been designated to be the parent of some enclave i at n. The constraints (4.3) require if relay r is controlled (indirectly through some substation) by enclave j in control center $P(P(r))$, then that enclave must be active. Next, the constraints (4.4) enforce that active enclaves have exactly one parent enclave and inactive enclaves have no parent. Similarly, the constraints (4.5) require every relay have exactly

one parent enclave. The constraints (4.6) require that if an enclave is designated as active, it must have at least one active child. Similarly, constraints (4.7) enforce that active control center enclaves have at least one grandchild relay. For each substation $s \in \mathcal{S}$ and enclave i in control center $P(s)$, the constraints (4.8) effectively define $z_{s,i}$ as an indicator of whether enclave i is a grandparent of some relay controlled by s. Constraint (4.9) enforces the investment budget governing how many enclaves can be added to balancing authorities and control centers, combined. Constraint (4.10) enforces the investment budgets regarding how many new enclaves can be added to substations. While the above constraints are enough to define the problem, we also include some constraints to reduce symmetry, namely

$$x_{n,i-1} \geq x_{n,i} \quad \forall n \in \mathcal{B} \cup \mathcal{C}, i \in E(N) \backslash \{0\}, \tag{4.14}$$

which requires that the active enclaves in control centers and balancing authorities be numbered sequentially starting at 0.

Given some fixed designer solution $\hat{y}$ and $\hat{q}$, the attacker's feasible region $\mathcal{A}(\hat{y}, \hat{q})$ is defined by the constraints:

$$\delta_r \leq \sum_{i \in E(P(P(r)))} \hat{q}_{r,i} \alpha_{P(P(r)),i} \qquad \forall r \in \mathcal{R} \tag{4.15}$$

$$\alpha_{n,i} \leq \sum_{j \in E(P(n))} \hat{y}_{n,i,j} \alpha_{P(n),j} \qquad \forall n \in \mathcal{C}, i \in E(n) \tag{4.16}$$

$$\beta_{s,i} \geq \hat{q}_{r,i} \delta_r \qquad \forall s \in \mathcal{S}, i \in E(P(s)), r \in C(s) \tag{4.17}$$

$$\hat{q}_{r,i} \beta_{s,i} \leq \delta_r \qquad \forall s \in \mathcal{S}, i \in E(P(s)), r \in C(s) \tag{4.18}$$

$$\sum_{s \in \mathcal{S}} \sum_{i \in E(P(s))} \beta_{s,i} + \sum_{n \in \mathcal{B} \cup \mathcal{C}} \sum_{i \in E(n)} \alpha_{n,i} \leq W \tag{4.19}$$

$$v_l \leq (1 - \delta_r) \qquad \forall l \in \mathcal{L}, r \in \mathcal{R}_l \tag{4.20}$$

$$v_l \geq \sum_{r \in \mathcal{R}_l} (1 - \delta_r) - |\mathcal{R}_l| + 1 \qquad \forall l \in \mathcal{L} \tag{4.21}$$

$$w_g \leq (1 - \delta_r) \qquad \forall g \in \mathcal{G}, r \in \mathcal{R}_g \qquad (4.22)$$

$$w_g \geq \sum_{r \in \mathcal{R}_g} (1 - \delta_r) - |\mathcal{R}_g| + 1 \qquad \forall g \in \mathcal{G} \qquad (4.23)$$

$$u_s \leq (1 - \delta_r) \qquad \forall s \in \mathcal{S}, r \in \mathcal{R}_s \qquad (4.24)$$

$$u_s \geq \sum_{r \in \mathcal{R}_s} (1 - \delta_r) - |\mathcal{R}_s| + 1 \qquad \forall s \in \mathcal{S} \qquad (4.25)$$

Recall that δ_r indicates whether the attacker compromises relay r, while $\alpha_{n,i}$ indicates whether he must attack enclave i in node n. Thus, the constraints (4.15) mean that in order for the attacker to compromise relay r, he must also compromise the control center enclave that is its ancestor. The constraints (4.16) require that to compromise an enclave in control center $n \in \mathcal{C}$, the attacker must compromise its parent enclave in the parent balancing authority $P(n)$. The variable $\beta_{s,i}$ denotes whether or not the attacker compromises the enclave in the substation s that is the child of enclave i in the substation's parent control center $P(s)$. The constraints (4.17) thus require that if any relay r that is a child of substation s and a descendant of enclave i in control center $P(s)$ is attacked, then the corresponding enclave in s must be attacked as well. The constraints (4.18) require that the attacker must attack all the relays he has access to by mandating that whenever he acquires access to some substation enclave, he must attack all its children. The attacker's budget is enforced by constraint (4.19). The constraints (4.20)-(4.25) translate between the set of relays compromised by the attack and the set of physical network components available after the attack. For example, the constraints (4.20) enforce that if a relay controlling line l is compromised, then line l is turned off. Conversely, constraints (4.21) enforce that whenever line l is off, at least one relay that controls it must be attacked.

Last, given fixed attacker decisions $\hat{u}$, $\hat{v}$, and $\hat{w}$, the defender's feasible region $\mathcal{X}(\hat{u}, \hat{v}, \hat{w})$ is defined by the constraints:

$$\sum_{l \in \mathcal{L}_s^+} f_l - \sum_{l \in \mathcal{L}_s^-} f_l + \sum_{g \in \mathcal{G}_s} p_g + \ell_s = D_s \qquad \forall s \in \mathcal{S} \qquad (4.26)$$

$$D_s(1 - \hat{w}_s) \leq \ell_s \leq D_s \qquad \forall s \in \mathcal{S} \tag{4.27}$$

$$-\overline{F}_l \hat{v}_l \leq f_l \leq \overline{F}_l \hat{v}_l \qquad \forall l \in \mathcal{L} \tag{4.28}$$

$$0 \leq p_g \leq \overline{P}_g \hat{u}_g \qquad \forall g \in \mathcal{G} \tag{4.29}$$

$$\ell_s \geq 0 \qquad \forall s \in \mathcal{S} \tag{4.30}$$

Flow balance at each substation is enforced by the constraints (4.26), where f_l is the directed flow on line l, p_g is the power generated at generator g, and ℓ_s is the load shed at substation s. The constraints (4.27)-(4.30) enforce variable bounds for the components still online after the attack, forcing all attacked components to be off.

4.3 Methodology

We solve the network segmentation model with an adaptation of the covering decomposition algorithm from Israeli and Wood (2002). We present our method in the form of two equivalent algorithms for solving a generic trilevel interdiction problem, and then discuss their application to our network segmentation problem (4.1) in Section 4.3.2. The general trilevel interdiction problem we consider is of the form

$$\nu := \min_{d \in \mathcal{D}} \left\{ \max_{a \in \mathcal{A}(d)} \left\{ \min_{x \in \mathcal{X}(a)} f(x) \right\} \right\}, \tag{4.31}$$

subject to the following assumptions:

Assumption 1. *We have relatively complete recourse at every level: That is, given $d \in \mathcal{D}$, one has $\mathcal{A}(d) \neq \emptyset$. Similarly, given any $d \in \mathcal{D}$ and $a \in \mathcal{A}(d)$, one has $\mathcal{X}(a) \neq \emptyset$.*

Assumption 2. *The set $\mathcal{D}$ is finite, as is the set $\mathcal{A}(d)$ for all $d \in \mathcal{D}$.*

Assumption 3. *The function f is bounded below by some number $\underline{u}$.*

While the relatively complete recourse assumption specifies that $\mathcal{A}(d)$ is nonempty for any $d \in \mathcal{D}$, it should be kept in mind that $\mathcal{A}(d)$ depends on d, so the designer may have the

ability to make any particular attack a infeasible. Let

$$\{a^1, a^2, \ldots, a^H\} = \bigcup_{d \in \mathcal{D}} \mathcal{A}(d) \tag{4.32}$$

and $\overline{K} = \{1, 2, \ldots, H\}$ be the corresponding index set for all possible attacks. Without loss of generality, we assume that there exists some fixed integer N such that the set $\mathcal{X}(a^k) \subseteq \mathbb{R}^N$ for all $k \in \overline{K}$. Then we may equivalently write (4.31) as

$$\nu = \min_{\substack{d \in \mathcal{D} \\ \xi \in \mathbb{R} \\ \eta \in \mathbb{R}^H \\ x^1, \ldots, x^H \in \mathbb{R}^N}} \quad \xi \tag{4.33}$$

$$\text{s.t.} \quad \xi \geq \eta_k \qquad \forall k \in \overline{K} \tag{4.34}$$

$$\begin{bmatrix} \eta_k = f(x^k) \\ x^k \in \mathcal{X}(a^k) \\ a^k \in \mathcal{A}(d) \end{bmatrix} \vee \begin{bmatrix} \eta_k = \underline{u} \\ a^k \notin \mathcal{A}(d) \end{bmatrix} \qquad \forall k \in \overline{K}. \tag{4.35}$$

By Assumption 2, $H < \infty$. Also by Assumption 2 and because the attacker problem is discrete, the constraints to require $a^k \notin \mathcal{A}(d)$ in the second disjunct in (4.35) define a discrete and finite set. Thus, it is possible at least in principle to solve (4.33)-(4.35), for example by introducing some additional variables to describe the disjunctions and expressing the problem as a mixed-integer program.

The key idea of this formulation is that by enumerating the second player's feasible set, we are able to combine the minimization operations of the first and third problems. The disjunctions in (4.35) represent the choice, for each second-player solution, to either find a third-player response (in the first disjunct), or for the first player to block the solution entirely (in the second disjunct). This disjunction is a consequence of the dependence between the second player's feasible region $\mathcal{A}(d)$ on the first player's decision d. The third-

player response x^k will be optimal for k where (4.34) is tight, since ξ is being minimized and does not appear in any other constraints. Thus, d will be an optimal solution to (4.31).

Next, consider replacing $\overline{K}$ with $K \subset \overline{K}$ in the formulation (4.33)-(4.35). Since this change removes some of the disjunctive constraints, the modified problem is a relaxation of the original one and provides a lower bound on ν. Incrementally adding attacks to K means incrementally adding disjunctive constraints, so it will result in a series of progressively tighter relaxations and hence monotonically nondecreasing lower bounds.

For practical applications, H will clearly be very large, meaning that (4.33)-(4.35) would have intractable numbers of constraints and variables. Instead of solving (4.33)-(4.35) directly, we propose an iterative algorithm that incrementally generates members of the set $\{a^1, a^2, \ldots, a^H\}$. This algorithm relies on being able to solve the bilevel problem formed by the inner two problems as a function of d:

$$u(d) = \max_{a \in \mathcal{A}(d)} \left\{ \min_{x \in \mathcal{X}(a)} f(x) \right\}. \tag{4.36}$$

While this problem may also be difficult, there are cases in which it may be tractable: for example, if the third-player problem is convex, then dualize-and-combine methods or the global Benders approach described in Wood (2011) are applicable.

For any feasible first-player solution d, the value $u(d)$ defined in (4.36) is an upper bound on the optimal objective value since it is the objective value of the second player's best response to a feasible first-player decision. The main idea of the algorithm is to alternate between solving (4.36) to get a second-player solution a^k, and thus an upper bound, and solving a relaxation of (4.33)-(4.35) that replaces $\overline{K}$ with the set K of indices of the second-player solutions enumerated so far. This relaxation of the original problem yields a lower bound and a new first-player solution to fix in (4.36). The details of the algorithm are given in Algorithm 2.

Algorithm 2: Solution algorithm with optimization-based designer

Input: An optimization problem of the form (4.31), as described by some representation of the set $\mathcal{D}$, the set functions $\mathcal{A}(\,\cdot\,)$ and $\mathcal{X}(\,\cdot\,)$, and a tolerance $\epsilon \geq 0$

Output: A first-player solution $d^* \in \mathcal{D}$ and corresponding objective value U such that that $U \leq \nu + \epsilon$, where ν is the optimal objective value of (4.31)

1 Let $k = 0$, $K = \emptyset$, $L = -\infty$, $U = \infty$
2 Select any initial first-player solution $d^0 \in \mathcal{D}$
3 **while** $U - L > \epsilon$ **do**
4 Fix $d = d^k$ in (4.36) and solve. Let $u(d^k)$ be the objective value and a^k be an optimal solution
5 **if** $u(d^k) < U$ **then**
6 $U \leftarrow u(d^k)$
7 $d^* \leftarrow d^k$
8 $K \leftarrow K \cup \{k\}$
9 Solve (4.33)-(4.35), letting the optimal solution be d^{k+1} and its optimal value be L.
10 $k \leftarrow k + 1$
11 **return** (d^*, U)

Based on our assumptions, Algorithm 2 terminates in finitely many iterations. Formally, we have the following result:

Proposition 3. *Under Assumptions 1-3, Algorithm 2 converges in finite time. The designer solution d^* it returns is ϵ-optimal for* (4.31)*, in the sense that its objective value U is no more than $\nu + \epsilon$.*

Proof. We claim that the algorithm cannot generate the same attack a^k twice, except perhaps in the final iteration. To see this, suppose that $0 \leq k_1 < k_2$ and $a^{k_1} = a^{k_2}$. At step 4 in iteration k_2, we then have $a^{k_1} = a^{k_2}$ being an optimal solution to (4.36) with $d = d^{k_2}$. So, attack a^{k_1} is not blocked by designer decision d^{k_2}, which was generated in the previous iteration $k_2 - 1 \geq k_1$. At iteration $k_2 - 1$, attack a^{k_1} can therefore not have been blocked and the first disjunct had to have been selected in (4.35) for $k = k_1 \in K$. So, after the solution of (4.33)-(4.35) in iteration $k_2 - 1$, one has $L = \xi \geq \eta_{k_1} = f(x^{k_1})$, where x^{k_1} optimizes f over $\mathcal{X}(a^{k_1})$. Returning now to iteration k_2, the algorithm inherits $L \geq f(x^{k_1})$ and by hypothesis regenerates a^{k_1} as the optimal solution to (4.36) in step 4. This means that

$u(d^{k_2}) = f(x^{k_1}) = f(x^{k_2})$, the minimum value of f over $\mathcal{X}(a^{k_1}) = \mathcal{X}(a^{k_2})$. Therefore, after steps 5-7, we will have $U \leq f(x^{k_1})$. The values of L produced in step 9 are monotonically nondecreasing, so upon reaching step 9 we will continue to have $L \geq f(x^{k_1})$ and also inherit $U \leq f(x^{k_1})$. Therefore, $U - L \leq 0 \leq \epsilon$ and the **while** loop will terminate after iteration k_2.

We conclude that the algorithm generates a new attack at each iteration in which it does not terminate. It can therefore run at most $H + 1$ iterations, since at that point there can be no new attacks to generate.

Concerning ϵ-optimality, each of the values $u(d^k)$ generated in step 4 is the objective value of an optimal attacker response to some feasible designer decision d^k, and is therefore an upper bound on ν as discussed earlier. At any iteration $\bar{k}$, the variable $U = \min_{k=0,\ldots,\bar{k}}\{u(d^k)\}$ is therefore also an upper bound on ν. As also discussed earlier, the value L produced at each iteration is the optimal value of a relaxation of (4.31), so we always have $L \leq \nu$. The algorithm only terminates when $U - L \leq \epsilon$, so we conclude that $U \leq L + \epsilon \leq \nu + \epsilon$ upon termination. □

Now, we show some properties of Algorithm 2 that will establish a connection to the covering decomposition algorithm of Israeli and Wood (2002).

Observation 2. *In Algorithm 2, $L = \underline{u}$ until the iteration the algorithm terminates, and the returned solution will be exactly optimal regardless of the value of ϵ.*

Proof. Consider an iteration $\bar{k}$ in which all the previous second-player solutions $\{a^1, a^2, \ldots, a^{\bar{k}-1}\}$ have been blocked at all prior iterations (that is, the second disjunct in (4.35) was selected). Then, after adding the solution $a^{\bar{k}}$, either it is possible to block all of $\{a^1, a^2, \ldots, a^{\bar{k}}\}$ — that is, find some $d^{\bar{k}} \in \mathcal{D}$ such that $a^1, \ldots, a^{\bar{k}} \notin \mathcal{A}(d^{\bar{k}})$ — or it is not. If so, the lower bound remains $\underline{u}$. If not, the attacks $\{a^1, a^2, \ldots, a^{\bar{k}}\}$ cannot all be blocked by one $d \in \mathcal{D}$, and the optimal objective value L of (4.33)-(4.35) at iteration $\bar{k}$ must be equal to $u(d^i)$ for some $i \in \{0, \ldots, \bar{k}\}$. Now, $U = u(d^*) \leq u(d^i)$ since it is the smallest optimal value seen

so far in step 4. But since $L = u(d^i)$, we conclude that $U \leq u(d^i) = L \leq L + \epsilon$, so the **while** loop in the algorithm will terminate following iteration $\bar{k}$ with an exact solution to the problem. □

Observation 2 yields a simpler algorithm: we need never consider the disjunction in (4.35), but may instead solve a feasibility problem including only the constraints in the second disjunct. That is, we find a solution d in the set defined by

$$\mathcal{D}(K) := \{d \in \mathcal{D} : a^k \notin \mathcal{A}(d) \, \forall k \in K\} \tag{4.37}$$

until $\mathcal{D}(K)$ is empty. The resulting algorithm is essentially the covering decomposition algorithm from Israeli and Wood (2002), treating the trilevel problem as a bilevel problem with a bilevel follower. It is presented, after some refinements described in the next section, as Algorithm 3 below.

It would have been possible to skip the presentation of Algorithm 2 and instead develop Algorithm 3 directly. We chose to start with Algorithm 2 for several reasons: first, we believe that the connection between the two algorithms is of interest and has not been clearly elucidated in the literature before. Second, there are straightforward modifications to Algorithm 2 that would still yield a correctly convergent method, but to which Observation 2 would no longer apply. In this case, the lower bounds L could potentially evolve in a nontrivial manner. Therefore, the value of ϵ might have a bearing on the algorithm output and it might be possible to terminate early with a suboptimal solution of known quality. Modifications to Algorithm 2 that could lead to such behavior include initializing K to be a nonempty set of attacks that are not simultaneously blockable, or supplementing the optimal solutions to the attacker problem (4.36) generated in step 4 with additional, suboptimal solutions. The practicality of such approaches would depend on the tractability of the designer's resulting disjunctive program (4.33)-(4.35). As we shall see in Section 4.4, however, the designer problem can be very challenging even in the simplified situation in which

the only goal is to block all the attacks known so far, so being able to solve (4.33)-(4.35) could well prove to be a major difficulty in using such possible extensions to Algorithm 2.

4.3.1 Some Variations on the Covering Decomposition Algorithm

Next, we make a new observation regarding the covering decomposition algorithm (Israeli and Wood, 2002): in Algorithm 2, it is sometimes unnecessary to solve (4.36) to optimality. Specifically, for iterations k in which the attacker problem (4.36) has an optimal value greater than or equal to the current upper bound U on the optimal value, it suffices in step 4 to compute any attack a^k inflicting damage of at least U on the designer solution d^k, as opposed to an attack with the greatest possible damage. Any such attack will fail the test $u(d^k) < U$ at step 5 in the same way as the optimal one, so U will not be updated. The finiteness and correctness of the algorithm only depend on a^k being an optimal attack in the iterations k in which U is updated, so neither is affected by this change. When solving (4.36) with a standard MIP solver, one may use a "cutoff" option to allow the solver to terminate early if it finds a feasible solution with objective value at least U. The modified algorithm is given in Algorithm 3. By Proposition 3, this algorithm returns an optimal first-player solution to (4.36) after a finite number of iterations.

Last, it is also possible to initialize the set K in Algorithm 3 as a set of precomputed second-player solutions, as long as the first-player problem is feasible for this set, that is, the first player can block all the solutions. While we leave this investigation for future work, it seems possible that a careful choice of initial attacks should reduce the number of iterations needed for convergence.

4.3.2 Applying the Algorithm for Power System SCADA Security

For our application, Algorithm 3 will fall victim to symmetry in the formulation of $\mathcal{D}(K)$ if multiple representations of equivalent first-player solutions do not block the same set of second-player solutions. For example, in the optimal network segmentation problem, we

Algorithm 3: Solution algorithm with feasibility-based designer

Input: An optimization problem of the form (4.31), as described by some representation of the set $\mathcal{D}$ and the set functions $\mathcal{A}(\,\cdot\,)$ and $\mathcal{X}(\,\cdot\,)$

Output: An optimal first-player solution $d^* \in \mathcal{D}$ to (4.31) and its objective value

Let $k = 0$, $K = \emptyset$, $U = \infty$
Select any initial first-player solution $d^0 \in \mathcal{D}$.
repeat
    Fix $d = d^k$ in (4.36) and solve. If the optimal value is at least U, any solution a^k with objective at least U may be returned; if the optimal objective value is below U, then let a^k be an optimal solution and $u(d^k)$ be the corresponding objective value
    if $u(d^k) < U$ **then**
        $U \leftarrow u(d^k)$
        $d^* \leftarrow d^k$
    $K \leftarrow K \cup \{k\}$
    Attempt to solve the feasibility problem (4.37), either finding some element $d^{k+1} \in \mathcal{D}(K)$ or determining that $\mathcal{D}(K) = \emptyset$
    $k \leftarrow k + 1$
until $\mathcal{D}(K) = \emptyset$
return (d^*, U)

define attacks to be the set of relays that are compromised, rather than the set of enclaves that the attacker infiltrates, as in Arguello et al. (2021). If attacks were to be represented as the set of compromised enclaves, Algorithm 3 would be guaranteed to enumerate through every possible relabeling of the enclaves before being able to fully block each attack.

The designer feasibility problem

With this representation in mind, we now present our formulation of the feasibility problem (4.37). This formulation introduces two additional families of variables: $t_{n,i,k}$ indicates whether enclave i in node n has to be compromised to execute attack k, and $\tau_{s,i,k}$ indicates whether any relay controlled by a substation s which is controlled by enclave i in its parent control center must be compromised to execute attack k. Next, for notational convenience, let

$$K(n) = \{k \in K : R(n) \cap A(k) \neq \emptyset\}, \tag{4.38}$$

that is, $K(n)$ is the set of attacks which compromise at least one relay descended from node n. Similarly, let

$$\mathcal{B}(k) = \{n \in \mathcal{B} : A(k) \cap R(n) \neq \emptyset\} \tag{4.39}$$

$$\mathcal{C}(k) = \{n \in \mathcal{C} : A(k) \cap R(n) \neq \emptyset\}. \tag{4.40}$$

$\mathcal{B}(k)$ and $\mathcal{C}(k)$ are the sets of balancing authorities and control centers, respectively, that must be compromised to execute attack k. Then, for a set of attacks K, the designer feasibility problem may be expressed by appending the following to (4.2)-(4.13):

$$t_{n,i,k} \leq \sum_{r \in R(n) \cap A(k)} q_{r,i} \qquad \forall n \in \mathcal{C}, i \in E(n), k \in K(n) \tag{4.41}$$

$$t_{n,i,k} \leq \sum_{v \in C(n) \cap \mathcal{C}(k)} \sum_{j \in E(v)} t_{v,j,k} y_{v,j,i} \qquad \forall n \in \mathcal{B}, i \in E(n), k \in K(n) \tag{4.42}$$

$$\tau_{s,i,k} \leq \sum_{r \in C(s) \cap A(k)} q_{r,i} \qquad \forall s \in \mathcal{S}, i \in E(P(s)), k \in K(s) \tag{4.43}$$

$$\sum_{s \in \mathcal{S}} \sum_{j \in E(P(s))} \tau_{s,j,k} + \sum_{n \in \mathcal{B}(k) \cup \mathcal{C}(k)} \sum_{i \in E(n)} t_{n,i,k} \geq W + 1 \qquad \forall k \in K \tag{4.44}$$

$$\tau_{s,i,k} \in \{0,1\} \qquad \forall s \in \mathcal{S}, i \in E(P(s)), k \in K(s) \tag{4.45}$$

$$t_{n,i,k} \in \{0,1\} \qquad \forall n \in \mathcal{C} \cup \mathcal{B}, i \in E(n), k \in K(n) \tag{4.46}$$

The constraints (4.41) enforce that, unless a control center enclave is an ancestor of a relay which is attacked in an attack k, that enclave is not necessary to accomplish attack k. Similarly, the constraints (4.42) enforce that a balancing authority enclave is not necessary for attack k unless one of its children is. While we write (4.42) in a nonlinear form, the nonlinearities are products of binary variables which we linearized via standard reformulations in our computational experiments. The constraints (4.43) effectively define $\tau_{s,i,k}$ to be an indicator of whether at least one relay controlled by substation s and attacked by k

has enclave i in $P(s)$ as an ancestor. The constraints (4.44) require that all attacks in K are over budget and therefore blocked.

Inner bilevel problem.

Ideally, the third-player re-dispatch problem would be as realistic a model of power grid operations as possible, suggesting an AC optimal power flow (ACOPF) formulation (Alderson et al., 2011). Unfortunately, the ACOPF equations are nonconvex (Carpentier, 1962), which would likely make (4.36) computationally prohibitive. For this reason, it is common to approximate ACOPF linearly, using the DCOPF formulation (O'Neill et al., 2011; Kocuk, Dey, et al., 2016). Though DCOPF is a linear program, Sundar, Coffrin, et al. (2018) and Chapter 3 of this thesis find that solving (4.36) via dualize and combine with DCOPF is still computationally challenging for large networks and large attacker budgets. In Chapter 3, we showed that in contexts where the objective is to minimize load shed and having correct line flows is not important, network flow is a good approximation of DCOPF. This holds with theoretical guarantees for uncongested networks, but was shown to hold empirically even for congested networks. So, as mentioned previously, we formulate (4.36) as the network flow restriction from Chapter 3, thereby producing a computationally tractable problem suitable for the dualize-and-combine approach. The resulting attacker MIP is given in formulation (F.15) in Appendix F.5, and we refer the reader back to Chapter 3 for details on the resulting single-level formulation.

4.4 Computational Results

In this section, we address two questions: First, what are the impacts of the attacker and defender budgets on the effectiveness of the optimal network segmentation? Second, what are the computational limitations of Algorithm 3 as applied to the network segmentation problem? To answer these questions, we ran Algorithm 3 on several test networks, each for a variety of attacker and defender budgets. We report both the objective value as a

percentage of the total load in the system and the computational time to solve the model. We show that, with well-chosen defense budgets, the damage from a cyberattack can be greatly reduced by optimal network segmentation, sometimes by as much as 50%. In addition, most of the improvement can be achieved via the combined balancing authority and control center defense budget. While increases in the substation budget sometimes further reduce the load shed of a worst-case attack, the reduction is usually moderate. However, increases in the substation budget greatly increase the computational time of Algorithm 3. Last, we show that, even for a realistically sized grid, we can use Algorithm 3 to solve network segmentation problems with large enough defender budgets to significantly reduce the impact of a worst-case attack that initially shed as much as 22% of the total system load.

We present results on three different power grid networks: the IEEE 30-bus case, a 500-bus network, and a 2,000 bus test system based on the ERCOT (Electric Reliability Council of Texas) network. For the 30-bus case, we use the communication network from Arguello et al. (2021). For the 500-bus case, we present results using two different communication networks: one has a single balancing authority and 5 control centers, and the other has 2 balancing authorities and 6 control centers. The substations are assigned to the control centers so that children of the same control center are nearby geographically. For the 2,000-bus system, we generated a control network with 2 balancing authorities and 5 control centers.

Our model is implemented in Pyomo (Hart et al., 2011; Bynum et al., 2021), and we use Gurobi version 9.0.2 to solve both the feasibility problem and the upper bounding problem (Gurobi Optimization, LLC, 2020). To solve the feasibility problem, we set Gurobi's `MIPFocus` parameter to 2 (prioritizing identification of feasible solutions) and cut off the solve after the first feasible solution. The objective (required by Gurobi) is to maximize the left-hand side of (4.44), with k corresponding to the most recently generated attack. We chose this objective since it rewards blocking the most recently generated attack; however,

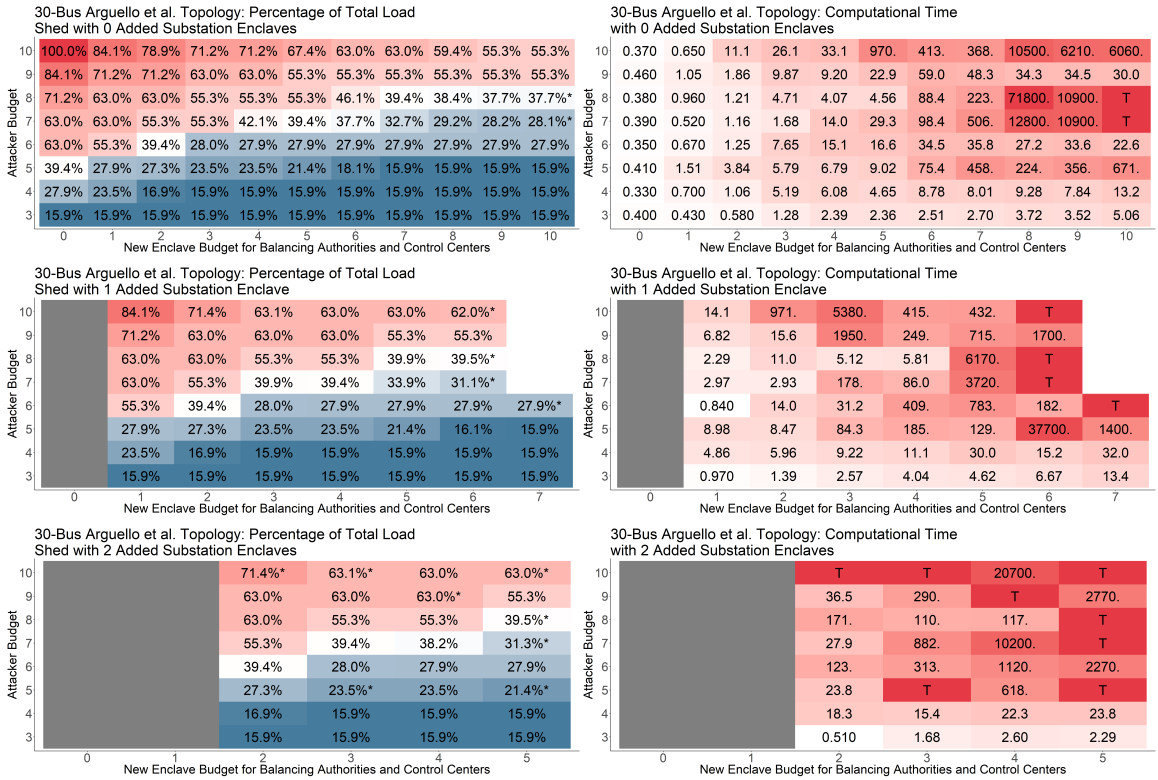


Figure 4.5: Computational results from running Algorithm 3 on the 30-bus test instance.

since we are mainly relying on Gurobi's primal heuristics, it is unclear how much the choice of objective matters. All experiments were run giving Gurobi 12 threads on a server with 40 Intel Xeon 2.20GHz CPUs and 256GB of RAM (our experience being that allocating more than 12 threads to Gurobi in this environment does not improve performance).

4.4.1 30-Bus Test Case

First, we discuss results for the 30-bus test system. Because of the small number of buses, we are able to solve instances of the network segmentation problem for attacker budgets of between 3 and 10 enclaves (attacking even a single relay requires compromising its substation, the substation's parent control center, and the control center's balancing authority, so the minimum budget to execute any attack is 3). Without any segmentation, an attacker budget of 10 is sufficient to shed all the load in the system. The results are depicted in Figure 4.5. We consider balancing authority and control center enclave budgets of between

0 and 10, and substation enclave budgets of 0, 1, and 2. In the first column of plots, we show the optimal objective value (or, for the ones marked with an asterisk, the best upper bound U in the case that Algorithm 3 does not converge within 24 hours) as a percentage of the total load in the system. The cells are colored from blue to white to red, where blue, white and red respectively correspond to the minimum, mean, and maximum percentage load shed across all the plots. In the second column, we show the computational time in seconds. The cells are colored according to the log of the computational time, from white to red, where white is the minimum run time in the table and red is 24 hours. We use 'T' to denote that Algorithm 3 did not converge within 24 hours. The first row of plots has a budget of 0 substation enclaves for the defender, the second row has a budget of 1 substation enclave, and the last row has a budget of 2 substation enclaves. Figures 4.6-4.8 use the same format.

We include fewer runs for the higher substation enclave budgets since they became computationally expensive. Note, however, that as the substation enclave budget increases, it is rare to get a large reduction in load shed. For example, between budgets of 0 and 1 substation enclaves, only 13 of the budget combinations benefit from the additional substation enclave. However, on average, the computational time increases by more than 2 hours. Comparing budgets of 1 and 2 substation enclaves, only 5 of the instances we solve see any improvement in load shed and the computational time increases by more than 6 hours on average.

Thus, for this test case, we find that most of the benefit from network segmentation is attained by investing solely in new enclaves in the balancing authority and control center levels. We can solve all but two of these instances within 24 hours. Load shed is typically reduced by about 50% as compared to the unsegmented network. The charts also indicate the trade-off between defender and attacker budgets: for any given attacker budget, there appears to be a point of diminishing returns after which more designer enclaves have little or no impact on total load shed; the location of this point increases with the attacker budget.

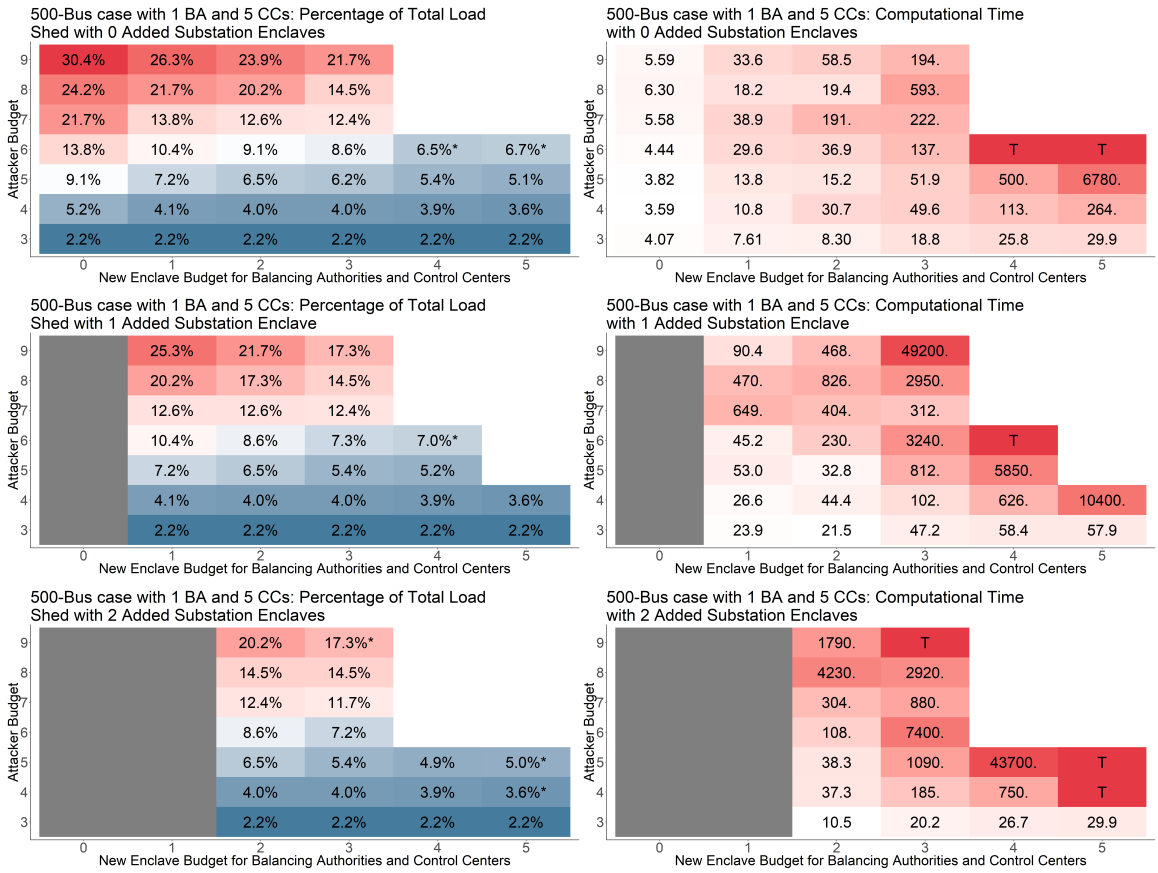


Figure 4.6: Computational results from running Algorithm 3 on the 500-bus test instance with a communication network with 1 balancing authority (BA) and 5 control centers (CCs).

4.4.2 500-Bus Test Cases

Next, we present experiments on two different control network structures for the 500-bus test case. Figure 4.6 shows results for the communication network with one balancing authority and 5 control centers. For the larger network, we are more computationally limited in what we can solve, but it is still possible to solve the network segmentation problem on attack budgets of up to 9 enclaves, which would shed 30% of the load without segmentation. Even a budget of 3 balancing authority and control center enclaves can reduce this load shed by about a third to about 22% of total system load. Again, we see limited utility in increasing the network designer's substation enclave budget. Between budgets of 0 and 1 substation enclaves, only 10 of the instances see a reduction in load shed with the additional

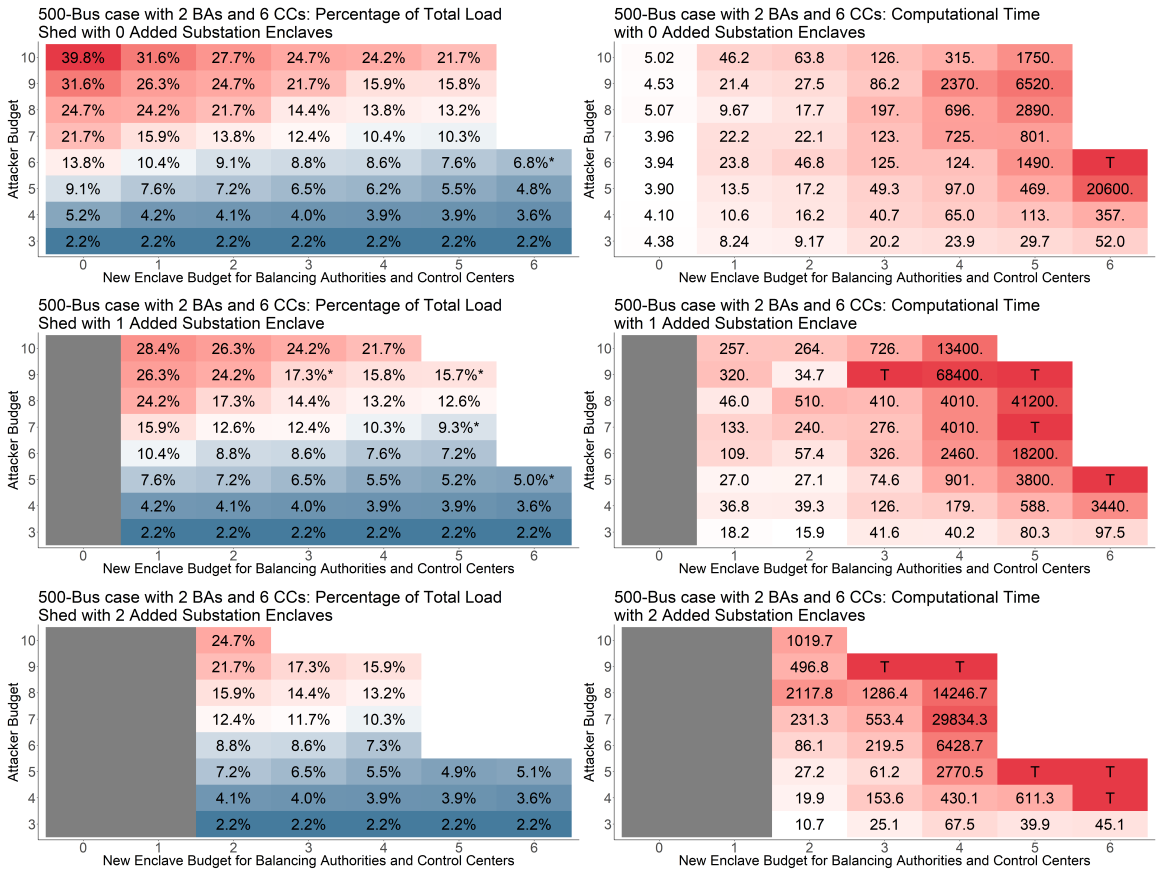


Figure 4.7: Computational results from running Algorithm 3 on the 500-bus test instance with a communication network with 2 balancing authorities (BAs) and 6 control centers (CCs).

substation enclave, and the reduction is at most 4.3% of the total load. Instances with larger attack budgets and larger budgets for balancing authorities and control centers benefit the most from having the additional substation enclave. Much as before, however, this comes at the cost of an additional 45 minutes of computational time, on average. Similarly, after increasing the substation enclave budget from 1 additional enclave to 2, only 7 instances have a reduction in load shed, by at most 2.8% of the total load. The average increase in computational time is more than 2 hours.

Next, Figure 4.7 depicts results for the same physical grid but with a different communication network topology having 2 balancing authorities and 6 control centers. The features observed in the previous two test cases are again present: the maximum benefit

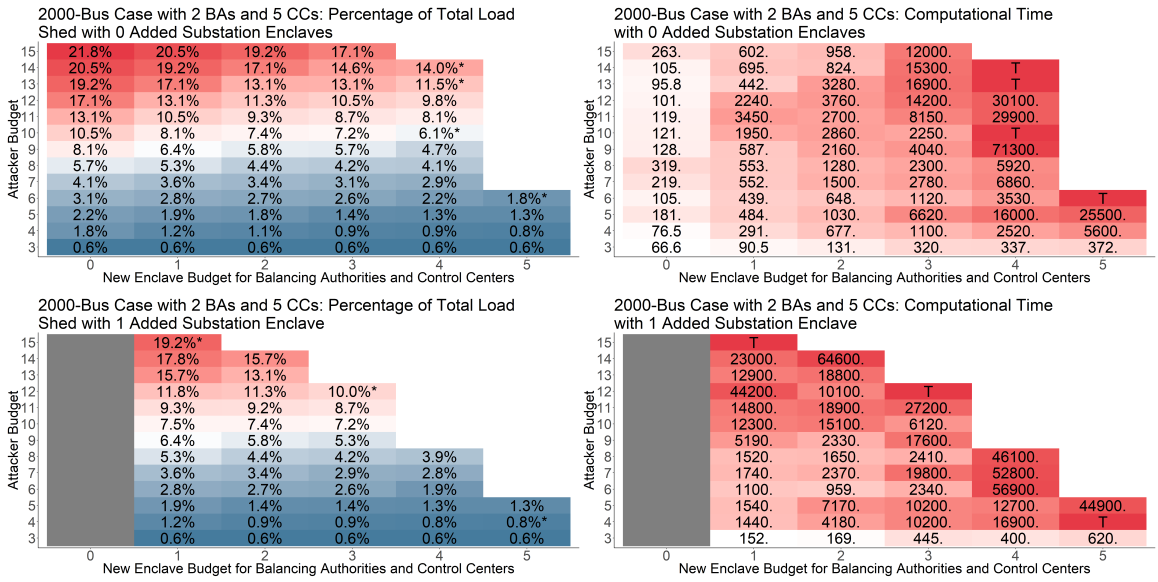


Figure 4.8: Computational results from running Algorithm 3 on the 2,000-bus test instance with a communication network with 2 balancing authorities (BAs) and 5 control centers (CCs).

from increasing the substation enclave budget from 0 to 1 enclave is again 4.3% of the total load, and only 20 of the 42 instances have any benefit. The computational time increases by more than 3 hours on average. Increasing from 1 substation enclave to 2 substation enclaves, 7 instances benefit in terms of load shed, by at most 2.6%. The computational time increases by more than 2 hours on average between these two sets of experiments.

4.4.3 2,000-Bus Test Case

Finally, Figure 4.8 depicts results on the 2,000-bus case based on the ERCOT grid, considering attacker budgets which can shed up to 22% of the total load in the absence of network segmentation. Computationally, these instances are much more challenging, but many are still solvable within 24 hours. Even more significantly than in the cases discussed previously, additional substation enclaves are of little benefit to the network designer. With an increase in the budget from 0 to 1, only 20 of the instances have decreased load shed, by at most 1.4% of the total load. To solve the instances with the larger substation enclave budget, however, takes 4.5 hours longer on average. For this test case, we did not perform

any experiments with a substation enclave budget of 2, since a budget of 1 was already very computational intensive and demonstrated only modest benefit. Overall, we observe that Algorithm 3 is capable of solving the network segmentation problem on a realistically sized grid, and for defender budgets that can decrease the load shed from a relatively severe worst-case attack by at least 20%, and sometimes considerably more: for example a defender budget of 4 can reduce the worst-case load shed from an attacker budget of 9 from 8.1% to 4.7%.

4.4.4 Performance Breakdown for Algorithm 3

While the previous section presented the total computational time for Algorithm 3, this section compares the time spent in the feasibility problem (the designer) and the upper-bounding problem (the attacker). As one might expect, the designer feasibility problem dominates the computational time for more challenging problems since it grows with each iteration. However, while it does not vary much over successive iterations, the attacker problem solution time can be significant for the larger test network.

Figure 4.9 shows plots of the computational time per iteration of Algorithm 3 for two different budget combinations on the 2,000-bus case. We subdivide the iteration time into the Gurobi solve times for the feasibility problem and the upper-bounding problem, as well as the time taken by overhead in our implementation. In the first plot, the attacker only has a budget of 5 enclaves, making the upper-bounding (attacker) problem relatively easy. Its time remains approximately constant as iterations increase, whereas the feasibility problem, plotted in the 'LB Solve Time' series tends to increase with the iterations. In contrast, in the second plot, the attacker has a higher budget, making the upper-bounding problem more difficult. In this run, the upper-bounding problem dominates the total time, and varies greatly between iterations. While the time for the feasibility problem increases slightly with the iterations, the time for the upper-bounding problem, while quite variable, does not appear much influenced by the iteration count.

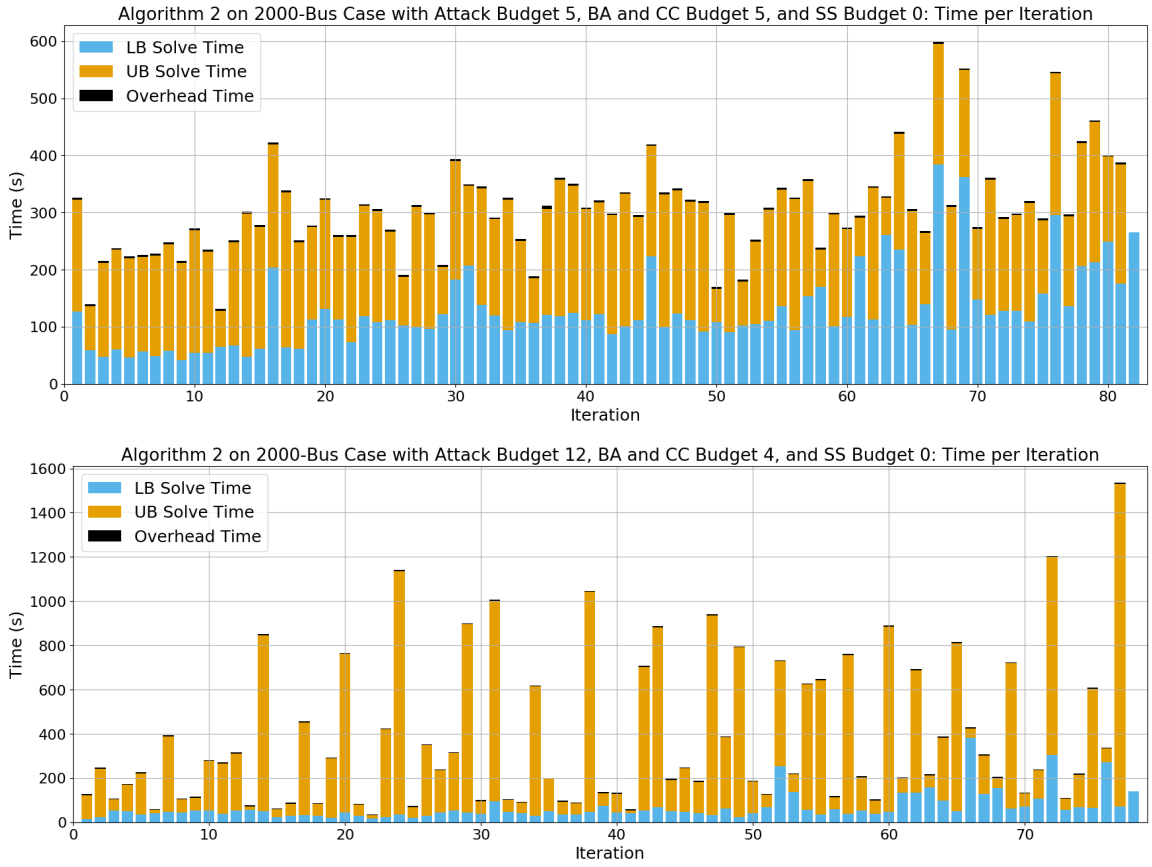


Figure 4.9: Computational time per iteration solving two different attacker and defender budget combinations on the 2,000-bus case. The first has a budget of 5 enclaves for the balancing authorities (BA) and control centers (CC), 0 enclaves for the substations (SS) and an attack budget of 5 enclaves. The second has a budget of 4 enclaves for the balancing authorities and control centers, no budget at the substation level, and an attack budget of 12 enclaves. 'LB Solve Time' and 'UB Solve Time' are the Gurobi solve times for the feasibility problem and the upper-bounding problem respectively. 'Overhead Time' is the time spent on overhead from our implementation.

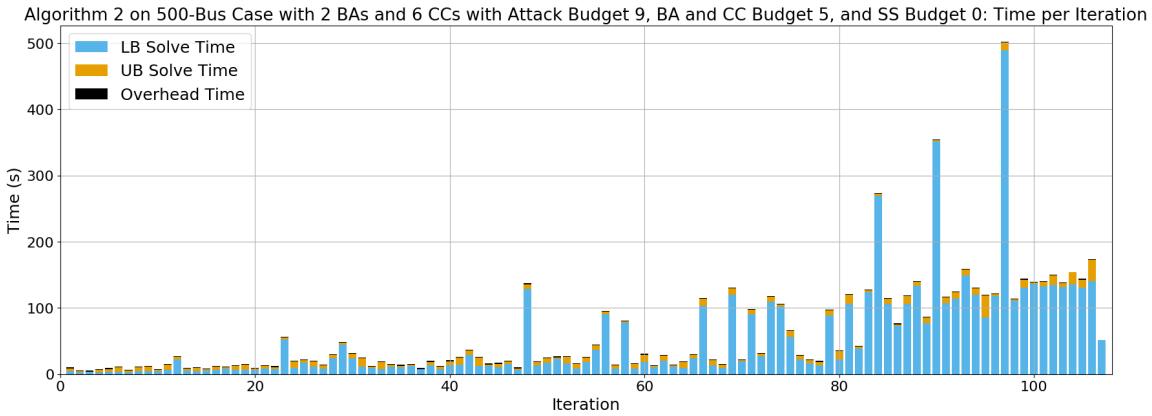


Figure 4.10: Computational time per iteration solving for a budget of 5 balancing authorities (BA) and control centers (CC), no additional substation (SS) enclaves, and an attack budget of 9 enclaves on the 500-bus case with 2 balancing authorities and 6 control centers. The color-coding scheme is the same as in Figure 4.9.

In contrast to the results in Figure 4.9, Figure 4.10 shows a plot of computational times per iteration for a particular budget on the 500-bus case with the 2-balancing-authority and 6-control-center SCADA network. In this plot, despite the attacker budget being relatively high for the instance, the total time is still dominated by the feasibility problem solution time. This outcome is unsurprising in light of the results from Chapter 3, since for a 500-bus network, we do not expect the upper-bounding problem to be difficult. Its time is nearly constant with iterations, whereas the time for the feasibility problem increases considerably.

Overall, we find that for smaller physical networks and for smaller attack budgets, the feasibility problem dominates the computational time in the algorithm. However, for large attack budgets on large physical networks, the upper-bounding problem can also be challenging, sometimes so much that it becomes dominant.

4.4.5 Benefit of Subproblem Cutoffs

This section explores the utility of using cutoffs when solving the upper-bounding problem (4.36). To do so, we make a comparison between Algorithm 3 as written and a version wherein we always solve the attacker problem to optimality. In Table 4.1, we compare

Table 4.1: Impact of the attacker problem objective cutoff described in Section 4.3.1. The boldface entries denote the version with fewer iterations in the "Number of Iterations" column and less time in the last two columns. Recall that V, T, and W are the substation enclave budget, balancing authority and control center enclave budget, and attacker budget respectively.

				Number of Iterations		Total Time (s)		Mean Problem (4.36) Time (s)	
Test Case	V	T	W	With Cutoff	Without Cutoff	With Cutoff	Without Cutoff	With Cutoff	Without Cutoff
30-bus	0	5	10	33	33	970.4	**844.9**	0.1	0.1
	0	7	7	**28**	29	506.3	**451.5**	0.2	0.2
	0	9	5	**35**	38	**355.9**	403.1	0.2	0.2
	0	9	10	34	**29**	6206.2	**1427.6**	0.1	0.1
	1	3	6	58	**30**	31.2	**11.1**	0.1	0.1
	1	4	7	**42**	53	**86.0**	185.9	0.1	0.1
	2	2	8	**113**	149	**171.5**	324.1	0.1	0.1
500-bus, 1BA5CC	0	2	9	**11**	16	**58.5**	135.1	**2.9**	6.1
	0	3	7	29	**21**	221.9	**186.2**	**4.6**	5.0
	0	4	5	**52**	54	**500.2**	575.0	**2.7**	3.1
	1	2	6	**33**	35	230.0	**162.0**	5.9	**3.7**
	2	3	5	**153**	205	**1087.7**	1863.5	**2.9**	3.4
	1	4	5	**160**	170	5846.6	**3394.2**	6.1	**3.4**
	2	2	6	26	**22**	**108.2**	125.3	**3.2**	5.2
	2	4	4	87	**79**	750.4	**671.8**	**1.8**	1.9
	1	2	8	67	**49**	826.3	**397.9**	11.2	**7.2**
500-bus, 2BA6CC	0	3	8	**34**	38	**197.1**	278.2	**3.5**	4.0
	0	4	10	36	**31**	315.1	**276.6**	**3.6**	3.8
	1	2	7	54	**47**	239.8	**199.5**	**3.0**	3.1
	1	4	5	85	**70**	900.8	**645.2**	3.0	**2.8**
	2	3	7	115	**102**	**553.4**	560.5	**2.8**	3.2
	0	4	9	**90**	150	**2372.3**	6381.9	**3.9**	4.8
	1	4	8	**156**	167	4014.5	**3456.6**	3.8	**3.4**
	2	3	8	148	**111**	1286.4	**708.0**	3.5	3.5
	2	2	8	350	**342**	**2117.8**	2150.6	**3.5**	3.8
	1	2	8	**95**	98	509.6	**478.0**	3.7	**3.6**
2000-bus	0	1	12	17	**15**	2236.7	**1933.3**	**136.8**	138.0
	0	2	8	4	4	**1278.3**	1399.3	**362.7**	397.2
	0	2	11	**13**	21	**2700.3**	6287.1	**227.8**	305.3
	0	3	6	5	5	**1115.5**	1272.8	**199.4**	226.5
	0	4	6	14	**13**	3525.1	**3317.5**	**209.2**	211.5
	1	1	8	8	8	**1523.2**	2083.6	**189.0**	259.9
	1	1	9	33	**24**	5186.3	**3678.2**	156.5	**153.4**
	1	2	6	6	6	**958.6**	1288.1	**149.5**	205.9
	1	2	9	**12**	14	**2333.4**	3466.6	**204.5**	251.3
	1	10	4	3	3	**695.2**	939.6	**206.7**	261.2

these variations for various budgets on each of the test cases. For each instance, using cutoffs or not, the table shows the number of iterations Algorithm 3 takes to converge, the total computational time for Algorithm 3, and the average time per iteration spent solving problem (4.36).

For all but the 2,000-bus case, using cutoffs has mixed results: the number of iterations can increase or decrease, as can the total computational time. The first use of a suboptimal attack can make the algorithm follow a different path, which can impact the number of iterations, the total time, and the difficulty of subsequent attacker and designer problems. For 30 or 500 buses, the designer feasibility problem dominates the computational time, so the number of iterations is the principal factor determining total run time, and total iterations respond somewhat unpredictably to the use of cutoffs.

In contrast, for the 2,000 bus case, the upper-bounding problem is more difficult. As we see in the 'Mean Problem (4.36) Time' columns, using cutoffs does tend to decrease the time spent in the upper-bounding problem for this case, decreasing the total computational time for Algorithm 3 with only a few exceptions. Therefore, it seems that for large physical networks where solving the worst-case attack problem is computationally challenging, the opportunity to terminate the attacker solve early can pay off in the overall computational time.

4.5 Conclusions and Future Work

We conclude that our algorithms make trilevel optimization network segmentation models appear computationally tractable for realistically sized power networks, at least for modest attacker and defender budgets. This may be a significant contribution, since network segmentation approaches capture the properties of real cyberattacks better than traditional models in which the designer can make particular physical assets invulnerable to any attack.

In reality, designer and attacker budgets will not simply be given and fixed. One may instead view the designer as having a decision problem with three competing objectives:

minimize the number of enclaves that need to be introduced, maximize the number of enclaves an attacker needs to penetrate, and contain load shed. Charts such as those in Figures 4.5-4.8 could help a designer decide how many enclaves to create and what kind of benefit to expect from them, and the solutions to the underlying problems could guide the designer in locating the enclaves.

In future work, enhancements to the solution techniques for solving both the feasibility problem and the attacker problem could significantly reduce the time for Algorithm 3 to converge and thus broaden the range of designer and attacker budgets that are solvable. In particular, the constraints (4.14) only partially remove the possible symmetries in describing equivalent designer decisions, so eliminating further symmetries might be very beneficial in solving instances with larger budgets. Another possibility that might bear some investigation is solving the designer feasibility problems by some other means than a standard MIP solver, for example, using a constraint logic programming solver. Also, techniques such as Benders decomposition might be applicable to the worst-case attack problem on large physical grids (Wood, 2011). Last, based on our observation that investing in substation enclaves is rarely beneficial, it may be possible to combine the two designer investment budgets. Then, the search for good designs could be guided, either formally or heuristically, by prioritizing investments at the balancing authority and control center levels before considering additional substation enclaves. Additional constraints, such as requiring that the substations controlled by a control center enclave be a connected component of the physical grid could also limit the search space, though perhaps at the cost of complicating the designer problem.

If the disjunctive lower-bounding problem (4.33)-(4.35) can be made tractable, variations on Algorithm 2 in which the designer problem is a disjunctive optimization problem instead of a feasibility problem could prove useful on the network segmentation problem and other trilevel interdiction problems. One advantage the disjunctive optimization ap-

proach might be the ability to find nontrivial lower bounds and hence meaningful optimality gaps that could allow for early termination with an approximate solution of known quality.

MIP solvers with more effective parallelism, which we expect will eventually be developed, might also be very beneficial to our algorithms.

4.6 Acknowledgments

The work in this chapter was in collaboration with Santanu Dey, Jonathan Eckstein, Cynthia Phillips, and John Siirola. We would also like to thank Bryan Arguello and Jared Gearhart for helpful conversations and help with computational experiments during this work.

This work was supported by Sandia National Laboratories' Laboratory Directed Research and Development (LDRD) program. Sandia National Laboratories is a multimission laboratory managed and operated by National Technology and Engineering Solutions of Sandia, LLC, a wholly owned subsidiary of Honeywell International, Inc., for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-NA0003525. SAND NO. XXXX-XXXX. The views expressed in the article do not necessarily represent the views of the U.S. Department of Energy or the United States Government.

Appendices

APPENDIX A

DCOTS MODEL

We use the following single-period formulation for DCOTS:

$$\min \quad \sum_{g\in\mathcal{G}} C_g p_g + R\sum_{s\in\mathcal{S}}(\ell_s + o_s) \tag{A.1}$$

$$\text{s.t.} \quad f_l \le B_l(\theta_{o(l)} - \theta_{d(l)}) + 2\pi B_l(1-y_l) \qquad \forall l\in\mathcal{L} \tag{A.2}$$

$$f_l \ge B_l(\theta_{o(l)} - \theta_{d(l)}) - 2\pi B_l(1-y_l) \qquad \forall l\in\mathcal{L} \tag{A.3}$$

$$\sum_{g\in\mathcal{G}_s} p_g + \sum_{l\in\mathcal{L}_s^+} f_l - \sum_{l\in\mathcal{L}_s^-} f_l = D_s - \ell_s + o_s \qquad \forall s\in\mathcal{S} \tag{A.4}$$

$$-\overline{F}_l y_l \le f_l \le \overline{F}_l y_l \qquad \forall l\in\mathcal{L} \tag{A.5}$$

$$\theta_{o(l)} - \theta_{d(l)} \ge -\frac{\pi}{6} - 2\pi(1-y_l) \qquad \forall l\in\mathcal{L} \tag{A.6}$$

$$\theta_{o(l)} - \theta_{d(l)} \le \frac{\pi}{6} + 2\pi(1-y_l) \qquad \forall l\in\mathcal{L} \tag{A.7}$$

$$\sum_{l\in\mathcal{L}}(1-y_l) \le K \tag{A.8}$$

$$\underline{P}_g \le p_g \le \overline{P}_g \qquad \forall g\in\mathcal{G} \tag{A.9}$$

$$-\pi \le \theta_s \le \pi \qquad \forall s\in\mathcal{S} \tag{A.10}$$

$$\ell_s, o_s \ge 0 \qquad \forall s\in\mathcal{S} \tag{A.11}$$

$$y_l \in \{0,1\} \qquad \forall l\in\mathcal{L} \tag{A.12}$$

This is the model used in Fisher et al. (2008) with the addition of load shed and over-generation so that the model is feasible for all topologies y. Generation costs are minimized in the first term of (A.1), and the second term penalizes infeasibility in the nodal balance constraints. We set R to be 10^6, several orders of magnitude higher than the maximum generation cost. Equations (A.2) and (A.3) are the McCormick relaxation of Ohm's law.

The nodal balance constraints with slacks included on the right-hand side are given in (A.4). Equation (A.5) sets the line flow to 0 when the line is opened and enforces transmission limits when the line is closed. In (A.6) and (A.7), we bound the phase angle differences when the line is closed. This is to help the accuracy of the DC approximation. We enforce a maximum cardinality of lines which can be opened in constraint (A.8). In practice, we expect the value of K to be 5 or 10. This is because switching large numbers of lines at once, even in large networks, makes finding an AC feasible dispatch difficult (Coffrin, Hijazi, et al., 2014). In addition, most of the cost savings from switching can be attained by switching a small number of lines (Kocuk, Jeon, et al., 2016). Equations (A.9)-(A.12) enforce variable bounds. Note that when the transmission switching binaries y_l are fixed, this formulation is the B-θ formulation for DCOPF, a linear program.

APPENDIX B

HEURISTIC ALGORITHMS FOR TRANSMISSION SWITCHING

In this appendix, we give formal descriptions of first the KNN heuristic algorithm and then the Local Search algorithm.

Algorithm 4: KNN for Transmission Switching

Input: Set of solved instances $\mathcal{Q} = \{I(q^1), I(q^2), \ldots, I(q^n)\}$ with ϵ-optimal transmission switching solutions $\{y^1, y^2, \ldots, y^n\}$, $p \geq 1$, $k \in \mathbb{Z}^+$, and an unsolved instance $I(q^{n+1})$

Output: Heuristic solution y^{n+1} for instance $I(q^{n+1})$

$T \leftarrow \emptyset$

let $\hat{q}^{n+1} = \frac{1}{||q^{n+1}||_2} q^{n+1}$

for $I(q^i) \in \mathcal{Q}$ **do**

    let $\hat{q}^i = \frac{1}{||q^i||_2} q^i$

    let $d_i = ||\hat{q}^{n+1} - \hat{q}^i||_p$

    if $|T| < k$ **then**

        $T \leftarrow T \cup \{I(q^i)\}$

    else

        for $I(q^j) \in T$ **do**

            if $d_i < d_j$ **then**

                $T \leftarrow (T \setminus \{I(q^j)\}) \cup \{I(q^i)\}$

LB $\leftarrow \infty$

for $I(q^j) \in T$ **do**

    Fix the transmission switching solution y^j in $I(q^{n+1})$ and solve.

    Let the optimal value be v_j

    if $v_j <$ *LB* **then**

        LB $\leftarrow v_j$

        $y^{n+1} \leftarrow y^j$

Algorithm 5: Greedy Local Search

Input: DCOTS instance $I(q)$, maximum number of lines that can be opened, K

Output: Heuristic solution y for instance $I(q)$

$S \leftarrow \mathcal{L}$

$\kappa \leftarrow 0$

Solve $I(q)$ with all lines closed. Let UB be the optimal value.

while $\kappa < K$ **do**

    $x \leftarrow \infty$

    for $l \in S$ **do**

        In $I(q)$, fix $y_l = 0$ and $y_k = 1$ for all $k \in S \setminus \{l\}$

        Solve $I(q)$ and let v be the optimal value.

        if $v < x$ **then**

            $x \leftarrow v$

            $m \leftarrow l$

    if $x \geq UB$ **then**

        STOP

    else

        Fix $y_m = 0$

        UB $\leftarrow x$

        $S \leftarrow S \setminus \{m\}$

        $\kappa \leftarrow \kappa + 1$

APPENDIX C

CONGESTION OF THE TRANSMISSION SWITCHING TEST INSTANCES

In this appendix, we give further detail on our seven test instances with regards to their congestion and hence their potential cost benefit from transmission switching.

In Table C.1, we show the average relative gap of the solution where no lines are opened for the 30 test instances on each of our test systems. Note that, for the 300kocuk, 1951rte_api, 2869pegase_api, and 3375wp_api cases, many of our training instances are not feasible with all lines closed: They require switching for feasibility, so part of the gap reported in the table is the infeasibility cost in our model. Also note that, for many instances, there are no additional savings for higher-cardinality switching budgets. This is unsurprising in light of the observation in Kocuk, Jeon, et al. (2016) that few lines are required to be switched in order to get most of the cost benefit from transmission switching. The 118blumsack and 3375wp_api networks are so congested they will switch more than 10 lines if allowed. However, note that, even in these cases, most of the cost savings can still be attained by switching only 5 lines.

In Figure C.1 we show the percentage of lines which have flow equal to the transmission limit, which is another indication of the congestion of the system. While we expect that more congestion implies more cost savings, note that this is not always the case, since Figure C.1 does not take generation costs into account: It is possible to have high numbers of congested lines but still not have as much cost benefit from transmission switching. For example, the 2869pegase_api case does not have a high percentage of tight lines, though it does get relatively high cost benefit from switching in Table C.1.

Table C.1: Average relative gap of the solution with no lines switched open compared to the best known solution. Note that for the instances marked with a "*", the gaps are above 100% because they are not always feasible with all lines on, so the DCOPF problem is paying an infeasibility cost for load shed. Also note that, in the 1354pegase case, we would expect the relative gap in the no-cardinality case to be the same or better than with cardinality 10. However, since the best known solutions from among all the heuristics are not all optimal, we see a slight error here.

Test Case	Cardinality 5	Cardinality 10	No Cardinality Constraint
118blumsack*	1118.00%	1234.00%	1248.00%
300kocuk*	365.00%	371.00%	371.00%
1354pegase	1.11%	1.12%	1.11%
1951rte_api	5.86%	5.89%	5.89%
2869pegase	0.35%	0.35%	0.35%
2869pegase_api	7.63%	7.67%	7.68%
3375wp_api	2.84%	3.03%	3.56 %

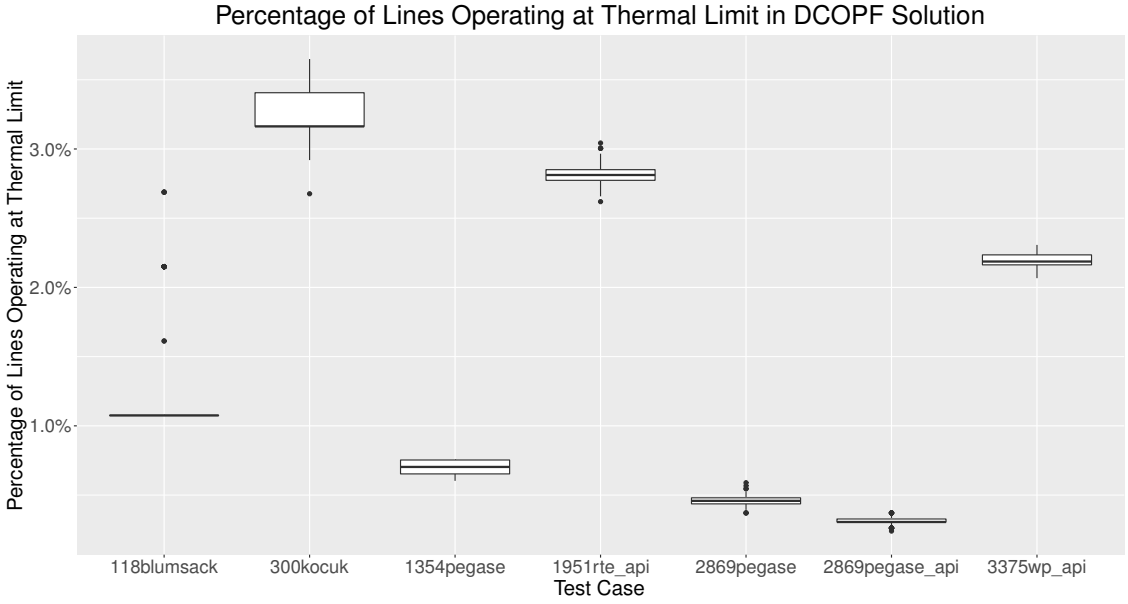


Figure C.1: Percentage of lines in each test system which have flow equal to the transmission limits when switching is not allowed for the 300 generated instances on each network.

APPENDIX D

PROOF OF PROPOSITION 1

Note that $\{0,1\}^n$ is a finite set, so we can rewrite (2.1) in the following way:

$$g(b) = \min_{x \in \{0,1\}^n} f_x(b), \tag{D.1}$$

where

$$\begin{aligned} f_x(b) = \min_{y} \quad & c_x^T x + c_y^T y \\ \text{s.t.} \quad & Gy \geq b - Ax \\ & y \in \mathbb{R}^m \end{aligned} \tag{D.2}$$

Note that (D.2) is a linear program, so we know that $f_x(b)$ is piecewise linear and convex in b. (We can assume that $f_{\hat{x}}(b) = +\infty$ if (D.2) is infeasible when the integer variables are fixed to $\hat{x}$.) This means that $g(b)$ is the minimum of finitely many piecewise linear convex functions. Fix some b and suppose that (x^*, y^*) is an optimal solution to (D.1) which is unique in the integer component.

Pick any other binary vector $\hat{x}$. We will show that there is a neighborhood $N(\hat{x})$ of b such that

$$f_{x^*}(\hat{b}) \leq f_{\hat{x}}(\hat{b}) \; \forall \hat{b} \in N(\hat{x}). \tag{D.3}$$

Proving this completes the proof since there are only a finite number of binary vectors, i.e., the final neighborhood we obtain will be the intersection of the neighborhoods corresponding to different binary variables.

There are two cases to consider:

1. $f_{\hat{x}}(b) = +\infty$: Since the set of right-hand sides for which an LP is feasible forms a polyhedron, i.e. a closed set, this implies there is a neighborhood of b where $f_{\hat{x}}(b) = +\infty$. Pick this neighborhood as $N(\hat{x})$.

2. $f_{\hat{x}}(b) < +\infty$: Note that $f_{\hat{x}}(b)$ is bounded from below since we have assumed that $f_{x^*}(b)$ is the global minimum. This implies that:

   (a) There exists a neighborhood around b such that both the functions $f_{\hat{x}}(\cdot)$ and $f_{x^*}(\cdot)$ are continuous in this neighborhood. (This follows from the fact that these functions are convex and convex functions are continuous in the relative interior of their domain.)

   (b) There exists $\epsilon > 0$ such that $f_{x^*}(b) \leq f_{\hat{x}}(b) + \epsilon$.

   The above two points indicate that there is a neighborhood around b such that (D.3) holds. □

APPENDIX E

SINGLE-LEVEL FORMULATIONS OF THE WORST-CASE RELAY ATTACK PROBLEM

In this appendix, we give the two single-level formulations of (3.1) that we compare against problem (3.10). Let ξ^+ and ξ^- represent the duals of constraints (3.1m) and (3.1n) respectively, and κ^+ and κ^- be the duals of the phase angle bound constraints in (3.1t). Then we have the logical formulation:

$$\max \quad \begin{array}{l} -\sum_{l\in\mathcal{L}} \left[F_l(\overline{\lambda}_l^+ + \overline{\lambda}_l^-) + 2\pi B_l(\overline{\xi}_l^+ + \overline{\xi}_l^-)\right] \\ -\sum_{g\in\mathcal{G}} \overline{P}_g\overline{\gamma}_g + \sum_{s\in\mathcal{S}} \left[D_s(\overline{\varphi}_s + \mu_s - \psi_s) - \pi(\kappa_s^+ + \kappa_s^-)\right] \end{array} \tag{E.1a}$$

s.t. (3.1b)-(3.1l)

$$\lambda_l^+ - \lambda_l^- + \mu_{d(l)} - \mu_{o(l)} + \xi_l^+ - \xi_l^- = 0 \qquad \forall l \in \mathcal{L} \quad (f) \tag{E.1b}$$

$$\mu_{s(g)} - \gamma_g \leq 0 \qquad \forall g \in \mathcal{G} \quad (p) \tag{E.1c}$$

$$\varphi_s + \mu_s - \psi_s \leq 1 \qquad \forall s \in \mathcal{S} \quad (\ell) \tag{E.1d}$$

$$\sum_{l\in\mathcal{L}_s^+} B_l(\xi_s^+ - \xi_s^-) + \sum_{l\in\mathcal{L}_s^-} (\xi_s^- - \xi_s^+) + \kappa_s^+ - \kappa_s^- = 0 \qquad \forall s \in \mathcal{S} \quad (\theta) \tag{E.1e}$$

$$u_s = 0 \implies \overline{\varphi}_s = \varphi_s \qquad \forall s \in \mathcal{S} \tag{E.1f}$$

$$u_s = 1 \implies \overline{\varphi}_s = 0 \qquad \forall s \in \mathcal{S} \tag{E.1g}$$

$$v_l = 0 \implies \overline{\xi}_l^+ = \xi_l^+ \qquad \forall l \in \mathcal{L} \tag{E.1h}$$

$$v_l = 1 \implies \overline{\xi}_l^+ = 0 \qquad \forall l \in \mathcal{L} \tag{E.1i}$$

$$v_l = 0 \implies \overline{\xi}_l^- = \xi_l^- \qquad \forall l \in \mathcal{L} \tag{E.1j}$$

$$v_l = 1 \implies \overline{\xi}_l^- = 0 \qquad \forall l \in \mathcal{L} \tag{E.1k}$$

$$v_l = 0 \implies \overline{\lambda}_l^+ = 0 \qquad \forall l \in \mathcal{L} \tag{E.1l}$$

$$v_l = 1 \implies \overline{\lambda}_l^+ = \lambda_l^+ \qquad \forall l \in \mathcal{L} \qquad \text{(E.1m)}$$

$$v_l = 0 \implies \overline{\lambda}_l^- = 0 \qquad \forall l \in \mathcal{L} \qquad \text{(E.1n)}$$

$$v_l = 1 \implies \overline{\lambda}_l^- = \lambda_l^- \qquad \forall l \in \mathcal{L} \qquad \text{(E.1o)}$$

$$w_g = 0 \implies \overline{\gamma}_g = 0 \qquad \forall g \in \mathcal{G} \qquad \text{(E.1p)}$$

$$w_g = 1 \implies \overline{\gamma}_g = \gamma_g \qquad \forall g \in \mathcal{G} \qquad \text{(E.1q)}$$

$$\xi_l^+ \geq 0 \qquad \forall l \in \mathcal{L} \qquad \text{(E.1r)}$$

$$\xi_l^- \geq 0 \qquad \forall l \in \mathcal{L} \qquad \text{(E.1s)}$$

$$\varphi_s \geq 0 \qquad \forall s \in \mathcal{S} \qquad \text{(E.1t)}$$

$$\psi_s \geq 0 \qquad \forall s \in \mathcal{S} \qquad \text{(E.1u)}$$

$$\lambda_l^+ \geq 0 \qquad \forall l \in \mathcal{L} \qquad \text{(E.1v)}$$

$$\lambda_l^- \geq 0 \qquad \forall l \in \mathcal{L} \qquad \text{(E.1w)}$$

$$\gamma_g \geq 0 \qquad \forall g \in \mathcal{G} \qquad \text{(E.1x)}$$

$$\kappa_s^+ \geq 0 \qquad \forall s \in \mathcal{S} \qquad \text{(E.1y)}$$

$$\kappa_s^- \geq 0 \qquad \forall s \in \mathcal{S} \qquad \text{(E.1z)}$$

We do not specify upper bounds on the dual variables of DCOPF and we encode constraints (E.1f)-(E.1q) using Gurobi `IndicatorConstraints`.

Next, we give a mixed integer linear programming reformulation of (E.1) with a heuristic upper bound on the dual variables. Let M represent the heuristic bound chosen for the dual variables of the DCOPF problem. We use this bound to give a mixed integer linear representation of the implications in (E.1f)-(E.1q):

$$\begin{aligned} \max \quad & -\sum_{l \in \mathcal{L}} \left[F_l(\overline{\lambda}_l^+ + \overline{\lambda}_l^-) + 2\pi B_l(\overline{\xi}_l^+ + \overline{\xi}_l^-) \right] \\ & -\sum_{g \in \mathcal{G}} \overline{P}_g \overline{\gamma}_g + \sum_{s \in \mathcal{S}} \left[D_s(\overline{\varphi}_s + \mu_s - \psi_s) - \pi(\kappa_s^+ + \kappa_s^-) \right] \end{aligned} \qquad \text{(E.2a)}$$

$$\text{s.t.} \quad \text{(3.1b)-(3.1l), (E.1b)-(E.1e), (E.1r)-(E.1z)}$$

$$0 \leq \overline{\varphi}_s \leq \varphi_s \qquad \forall s \in \mathcal{S} \qquad \text{(E.2b)}$$

$$\varphi_s - M u_s \leq \overline{\varphi}_s \leq M(1 - u_s) \qquad \forall s \in \mathcal{S} \qquad \text{(E.2c)}$$

$$0 \leq \overline{\xi}_l^+ \leq \xi_l^+ \qquad \forall l \in \mathcal{L} \qquad \text{(E.2d)}$$

$$\xi_l^+ - M v_l \leq \overline{\xi}_l^+ \leq M(1 - v_l) \qquad \forall l \in \mathcal{L} \qquad \text{(E.2e)}$$

$$0 \leq \overline{\xi}_l^- \leq \xi_l^- \qquad \forall l \in \mathcal{L} \qquad \text{(E.2f)}$$

$$\xi_l^- - M v_l \leq \overline{\xi}_l^- \leq M(1 - v_l) \qquad \forall l \in \mathcal{L} \qquad \text{(E.2g)}$$

$$0 \leq \overline{\lambda}_l^+ \leq \lambda_l^+ \qquad \forall l \in \mathcal{L} \qquad \text{(E.2h)}$$

$$\lambda_l^+ - M(1 - v_l) \leq \overline{\lambda}_l^+ \leq M v_l \qquad \forall l \in \mathcal{L} \qquad \text{(E.2i)}$$

$$0 \leq \overline{\lambda}_l^- \leq \lambda_l^- \qquad \forall l \in \mathcal{L} \qquad \text{(E.2j)}$$

$$\lambda_l^- - M(1 - v_l) \leq \overline{\lambda}_l^- \leq M v_l \qquad \forall l \in \mathcal{L} \qquad \text{(E.2k)}$$

$$0 \leq \overline{\gamma}_g \leq \gamma_g \qquad \forall g \in \mathcal{G} \qquad \text{(E.2l)}$$

$$\gamma_g - M(1 - w_g) \leq \overline{\gamma}_g \leq M w_g \qquad \forall g \in \mathcal{G} \qquad \text{(E.2m)}$$

APPENDIX F

PROOFS OF THEOREMS FROM CHAPTER 3

F.1 Proof of Observation 1.

Let I_k be the $k \times k$ identity matrix. Without loss of generality we may assume that there is exactly one generator per bus. We can do this because we already assumed the generator dispatch lower bound is 0, so if there are multiple generators at a bus, we can aggregate them into one by summing their maximum capacities. Note that this means $|\mathcal{G}| = |\mathcal{S}|$ in the following. To show the claim, we will show that the elements of any extreme point of the dual polyhedron, defined by (3.3b)-(3.3i), are bounded in absolute value by 1. For notational convenience, let $\{x : Gx \leq h\}$ be the system (3.3b)-(3.3i). Note that G is an integer matrix. Let $n = 3|\mathcal{S}|+2|\mathcal{L}|+|\mathcal{G}|$, the dimension of the dual space. Then an extreme point of the polyhedron is the feasible solution where a subsystem of n inequalities from $Gx \leq h$ hold at equality. Let $\overline{G}$ denote the square submatrix of G corresponding to this subsystem. By Cramer's Rule, this means that we can calculate the ith component of that solution:

$$\hat{x}_i = \frac{\det\left(\begin{bmatrix} \overline{G}^1 & \overline{G}^2 & \cdots & \overline{G}^{i-1} & h & \overline{G}^{i+1} & \cdots & \overline{G}^n \end{bmatrix}\right)}{\det(\overline{G})},$$

where $\overline{G}^j$ is the jth column of $\overline{G}$. Since G is integer, we know that $|\det(\overline{G})| \geq 1$. This means that

$$|\hat{x}_i| \leq \left|\det\left(\begin{bmatrix} \overline{G}^1 & \overline{G}^2 & \cdots & \overline{G}^{i-1} & h & \overline{G}^{i+1} & \cdots & \overline{G}^n \end{bmatrix}\right)\right|. \tag{F.1}$$

In the following, we show that the right-hand side of (F.1) is 1 by showing the matrix in question is totally unimodular.

We will show that $\begin{bmatrix} G & h \end{bmatrix}$ is totally unimodular since that means any submatrix of $\begin{bmatrix} G & h \end{bmatrix}$ is totally unimodular. Writing the columns corresponding to the ordering of the variables $(\lambda^+, \lambda^-, \mu, \gamma, \varphi, \psi)$, we can write

$$\begin{bmatrix} G & h \end{bmatrix} = \begin{bmatrix} I_{|\mathcal{L}|} & -I_{|\mathcal{L}|} & N^\top & 0 & 0 & 0 & 0 \\ -I_{|\mathcal{L}|} & I_{|\mathcal{L}|} & -N^\top & 0 & 0 & 0 & 0 \\ 0 & 0 & I_{|\mathcal{G}|} & -I_{|\mathcal{G}|} & 0 & 0 & 0 \\ 0 & 0 & I_{|\mathcal{S}|} & 0 & I_{|\mathcal{S}|} & -I_{|\mathcal{S}|} & \mathbf{1} \\ 0 & 0 & 0 & 0 & -I_{|\mathcal{S}|} & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & -I_{|\mathcal{S}|} & 0 \\ -I_{|\mathcal{L}|} & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & -I_{|\mathcal{L}|} & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & -I_{|\mathcal{G}|} & 0 & 0 & 0 & 0 \end{bmatrix} \tag{F.2}$$

where, without loss of generality, we relabel the generators so that we get the identity in the part of the matrix corresponding to μ in constraints (3.3c). We use N to represent the node-arc adjacency matrix of the network, which is known to be totally unimodular. We use $\mathbf{1}$ to represent the vector of all 1's in $\mathbb{R}^{|\mathcal{S}|}$. From (F.2), we see it suffices to show that

$$A = \begin{bmatrix} N^\top & 0 \\ I_{|\mathcal{G}|} & 0 \\ I_{|\mathcal{S}|} & \mathbf{1} \end{bmatrix}$$

is totally unimodular since $\begin{bmatrix} G & h \end{bmatrix}$ augments A by a series of identities and the negative of the first row. It is easy to verify that A is totally unimodular, for example it satisfies the conditions of the theorem by Hoffman (Heller and Tompkins, 1956).

By the definition of total unimodularity, it follows from the claim above and equation (F.1) that $-1 \leq \hat{x}_i \leq 1$ for all i. This means that 1 is a valid upper bound for φ, ψ, λ^+, λ^-, γ, and μ, and -1 is a valid lower bound for μ in problem (3.3). □

F.2 Proof of Theorem 1

We will first establish some lemmas before we give a proof of Theorem 1. Let N be the $|V| \times |E|$ node-arc incidence matrix of a connected digraph $G(V, E)$. We remind the reader of two facts:

- The rank of N is $|V| - 1$.
- Let $N^{(i)}$ be the matrix where the ith row if N is removed and let N_i be the ith row of N. Then

$$N_i = -\sum_j N_j^{(i)}. \tag{F.3}$$

  That is, we can calculate any given row of the matrix by taking the negative of the sum of the other rows.

In the following Lemma, we derive the injection shift factor formulation for DCOPF.

Lemma 1. *Consider a DCOPF problem over a connected digraph $G(V, E)$. Let $B \in \mathbb{R}^{|E|\times|E|}$ be a diagonal matrix where the diagonal entries are the susceptances of the lines. Let $0 \in V$ and consider the matrix $N^{(0)}$. Let $d \in \mathbb{R}^{|V|}$ be a set of injections which satisfy global balance, i.e.,*

$$\sum_j d_j = 0.$$

Let $d^{(0)}$ be the vector of injections where we have removed the component corresponding to the 0th node. Then the unique vector of flows that satisfies (i) nodal balance constraints given the injections d and (ii) Ohm's law is

$$f = B(N^{(0)})^\top \left(N^{(0)} B (N^{(0)})^\top\right)^{-1} d^{(0)}.$$

Proof. Let $\theta^{(0)}$ be the vector of phase angles where we have removed the component corresponding to the 0th node. Then f and $\theta^{(0)}$ must satisfy

$$N^{(0)} f = d^{(0)} \qquad \text{(nodal balance)} \tag{F.4}$$

$$f = B(-N^{(0)})^\top \theta^{(0)} \qquad \text{(Ohm's law)} \tag{F.5}$$

Combining (F.4) and (F.5),

$$-N^{(0)} B (N^{(0)})^\top \theta^{(0)} = d^{(0)}.$$

Since $N^{(0)}$ is full row rank and B is a diagonal matrix with all entries positive, $N^{(0)} B (N^{(0)})^\top$ is invertible. This means

$$\theta^{(0)} = -\left(N^{(0)} B (N^{(0)})^\top\right)^{-1} d^{(0)} \tag{F.6}$$

$$\Rightarrow f = B(N^{(0)})^\top \left(N^{(0)} B (N^{(0)})^\top\right)^{-1} d^{(0)}. \tag{F.7}$$

Set the phase angle of the 0th node to 0. Then by (F.3), the resulting phase angles and the vector of flows above satisfy nodal balance constraints and Ohm's law constraints. □

Given a square matrix H, let $\lambda_{\max}(H)$ be the largest eigenvalue of H.

Lemma 2. *Let $A \in \mathbb{R}^{m \times n}$ be a matrix with full row rank. Then $\lambda_{\max}(A^\top (AA^\top)^{-1} A) = 1$.*

Proof. The matrix $A^\top (AA^\top)^{-1} A$ is an orthogonal projection matrix, so all of its eigenvalues are 1 or 0 (since it is idempotent). Since A has rank m, so does $A^\top (AA^\top)^{-1} A$, so m of them are 1, and we have the result. □

Lemma 3. *Let $d \in \mathbb{R}^n$ be such that $\sum_{i=1}^n d_i = 0$. Then $|\sum_{i \in S} d_i| \leq \frac{1}{2}\|d\|_1$ for all $S \subseteq \{1, 2, \ldots n\}$*

Proof. The statement is trivially true for $S = \emptyset$ and $S = \{1, 2, \ldots, n\}$. So assume $\emptyset \subsetneq S \subsetneq \{1, 2, \ldots, n\}$. Note first that $\sum_{i \in S} |d_i| + \sum_{i \in \{1,2,\ldots,n\} \setminus S} |d_i| = \|d\|_1$. Then we have

that

$$\min\left\{\sum_{i\in S}|d_i|,\sum_{i\in\{1,2,\ldots,n\}\setminus S}|d_i|\right\}\leq\frac{1}{2}\|d\|_1. \tag{F.8}$$

By the triangle inequality we have,

$$\left|\sum_{i\in S}d_i\right|\leq\sum_{i\in S}|d_i|, \tag{F.9}$$

and since $\sum_{i=1}^{n}d_i=0$,

$$\left|\sum_{i\in S}d_i\right|=\left|\sum_{i\in\{1,2,\ldots,n\}\setminus S}d_i\right|\leq\sum_{i\in\{1,2,\ldots,n\}\setminus S}|d_i|. \tag{F.10}$$

Combining (F.9) and (F.10), we get

$$\left|\sum_{i\in S}d_i\right|\leq\min\left\{\sum_{i\in S}|d_i|,\sum_{i\in\{1,2,\ldots,n\}\setminus S}|d_i|\right\},$$

so the result follows from (F.8). □

We are ready to prove the theorem.

Proof of Theorem 1. Since d is flow-polytope feasible, let $f^{\text{nf}}\in\mathbb{R}^{|\mathcal{L}|}$ be the flow vector that satisfies thermal limits and nodal balance constraints given the node injection values d. We must show that there exists a flow vector that not only satisfies nodal balance constraints given the injections d and thermal limits, but also Ohm's law.

Claim 1: *It is sufficient to prove the DCOPF polytope is non-empty on each of the subgraphs corresponding to V^i, where we may assume that the ℓ_1-norm of the injections on the vertices V^i is at most $\|d\|_1$.*

Claim 1 is straightforward to verify, so we only sketch the arguments here. For the arcs connecting vertex blocks V^i and V^j where $i\neq j$, we will keep the flow values from f^{nf}. That flow clearly satisfies the thermal limit, and since those arcs are not involved in any cycles, once we find flow values on the incident arcs within each V^i, we will be able to find

values of θ such that Ohm's law will also be satisfied. Thus, the problem reduces to finding flows within blocks of nodes V^i for $i \in \{1, 2, \ldots, m\}$. It is straightforward then to show that the ℓ_1-norm of the injections on the vertices V^i is at most $\|d\|_1$.

Consider a block V (we drop the superscript for simplicity), recalling that it has at most $r(G)$ nodes. Let the net injections on the nodes be d^V such that $\|d^V\|_1 \leq \|d\|_1$. For simplicity of notation, we will refer to the subgraph on V as $H(V, E)$. Let $N \in \{0, 1, -1\}^{|V|\times|E|}$ be the node-arc incidence matrix of H. Let $B \in \mathbb{R}^{|E|}$ be a diagonal matrix with B_{ee} equal to the susceptance on arc e. Let $v_0 \in V$ be an arbitrarily chosen reference bus, let $N^{(v_0)}$ be as defined before, and let $d^{(v_0)}$ be the vector where we have removed the component corresponding to v_0 from d^V. Then by Lemma 1, the unique flow that satisfies the DCOPF constraints on block V is

$$f = B(N^{(v_0)})^\top \left(N^{(v_0)} B (N^{(v_0)})^\top\right)^{-1} d^{(v_0)}. \tag{F.11}$$

Let $\sqrt{B}$ be a diagonal matrix whose (e, e)th entry is $\sqrt{B_{ee}}$.

Claim 2: *There exists a vector $\tilde{d} \in \mathbb{R}^{|E|}$ such that*

$$(N^{(v_0)}\sqrt{B})\tilde{d} = d^{(v_0)}, \text{ and } \|\tilde{d}\|_2 \leq \frac{\sqrt{r(G)-1}}{2\sqrt{B_{\min}}}\|d^V\|_1.$$

We will show that there exists $x \in \mathbb{R}^{|E|}$ such that $N^{(v_0)}x = d^{(v_0)}$ and $\|x\|_2 \leq \frac{\sqrt{r(G)-1}}{2}\|d^V\|_1$. This completes the proof since we can then find $\tilde{d}$ by solving $\sqrt{B}\tilde{d} = x$. In the solution, we will have $\tilde{d} \leq \frac{1}{\sqrt{B_{\min}}}x$, since $\sqrt{B}$ is a diagonal matrix and $\sqrt{B_{\min}}$ is the smallest diagonal entry. This means that

$$\|\tilde{d}\|_2 \leq \frac{1}{\sqrt{B_{\min}}}\|x\|_2 \leq \frac{\sqrt{r(G)-1}}{2\sqrt{B_{\min}}}\|d^V\|_1,$$

and $(N^{(v_0)}\sqrt{B})\tilde{d} = N^{(v_0)}x = d^{(v_0)}$, as required.

Let $N^{(v_0)} = \begin{bmatrix} P & Q \end{bmatrix}$ where P is composed of columns corresponding to the arcs of a spanning tree in $H(V, E)$. This means that P is a full row rank square matrix, and furthermore that it is totally unimodular since it is the adjacency matrix of a bipartite graph. Solve

$$P\hat{d} = d^{(v_0)} \tag{F.12}$$

and let $x = \begin{bmatrix} \hat{d} \\ 0 \end{bmatrix}$. So it is sufficient to show that $\|\hat{d}\|_2 \leq \frac{\sqrt{r(G)-1}}{2}\|d^V\|_1$. Note that, by (F.12), $\hat{d}$ is a flow on the tree corresponding to P where the injections on the nodes are given by $d^{(v_0)}$. Note that since P represents a tree, the removal of any arc of the graph disconnects the graph. If we remove arc i, let S_i represent the set of nodes in the component containing $o(i)$. Using this notation, this means that for all arcs i, $\hat{d}_i = \sum_{j \in S_i} d^V_j$. By Lemma 3, this means that $|\hat{d}_i| \leq \frac{1}{2}\|d^V\|_1$. Finally, the support of $\hat{d}$ is at most $r(G)-1$, since a tree on $r(G)$ nodes has $r(G) - 1$ arcs. So $\|\hat{d}\|_2 = \sqrt{\sum_{i=1}^{r(G)-1} \hat{d}_i^2} \leq \frac{\sqrt{r(G)-1}}{2}\|d^V\|_1$, showing Claim 2.

Now, we can rewrite (F.11) as

$$\begin{aligned}
f &= B(N^{(v_0)})^\top \left(N^{(v_0)} B (N^{(v_0)})^\top\right)^{-1} d^{(v_0)} \\
&= B(N^{(v_0)})^\top \left(N^{(v_0)} B (N^{(v_0)})^\top\right)^{-1} N^{(v_0)} \sqrt{B} \tilde{d} \\
&= \sqrt{B}\sqrt{B}(N^{(v_0)})^\top \left(N^{(v_0)} \sqrt{B}\sqrt{B} (N^{(v_0)})^\top\right)^{-1} N^{(v_0)} \sqrt{B} \tilde{d} \\
&= \sqrt{B}(N^{(v_0)}\sqrt{B})^\top \left((N^{(v_0)}\sqrt{B})(N^{(v_0)}\sqrt{B})^\top\right)^{-1} (N^{(v_0)}\sqrt{B})\tilde{d}
\end{aligned}$$

where the second equality holds by Claim 2 and the last holds since $\sqrt{B}$ is symmetric. Therefore

$$\begin{aligned}
\|f\|_2 &\leq \lambda_{\max}\left(\sqrt{B}(N^{(v_0)}\sqrt{B})^\top \left((N^{(v_0)}\sqrt{B})(N^{(v_0)}\sqrt{B})^\top\right)^{-1} (N^{(v_0)}\sqrt{B})\right) \|\tilde{d}\|_2 \\
&\leq \lambda_{\max}(\sqrt{B})\lambda_{\max}\left(N^{(v_0)}\sqrt{B})^\top \left((N^{(v_0)}\sqrt{B})(N^{(v_0)}\sqrt{B})^\top\right)^{-1} (N^{(v_0)}\sqrt{B})\right) \|\tilde{d}\|_2 \\
&\leq \sqrt{B_{\max}} \cdot 1 \cdot \frac{\sqrt{r(G)-1}}{2\sqrt{B_{\min}}}\|d^V\|_1,
\end{aligned}$$

where the last inequality follows from Lemma 2 and from Claim 2.

By the assumption of the theorem, $\overline{F}_e \geq \sqrt{\frac{B_{\max}}{B_{\min}}} \frac{\sqrt{r(G)-1}}{2} \|d\|_1$, for all $e \in E$. Thus the f above is feasible, completing the proof. □

F.3 Proof of Corollary 1

By construction, problems (3.13) and (3.14) are both bounded and feasible (since it is always possible to shed all the load and since ℓ is bounded). Also, since (3.14) is a relaxation of (3.13), $z^l \leq z^*$. It is sufficient to show that there exists a solution to (3.13) with the same objective value as (3.14).

Let $(\tilde{f}, \tilde{\ell}, \tilde{p})$ be an optimal solution to (3.14). Since $0 \leq \tilde{\ell} \leq D$ and $0 \leq \tilde{p} \leq \overline{P}$, it is sufficient to show that the system

$$\begin{aligned}
\sum_{l \in \mathcal{L}_s^+} f_l - \sum_{l \in \mathcal{L}_s^-} f_l &= D_s - \tilde{\ell}_s - \sum_{g \in \mathcal{G}_s} \tilde{p}_g \quad \forall s \in \mathcal{S} \\
f_l &= B_l(\theta_{o(l)} - \theta_{d(l)}) \qquad \forall l \in \mathcal{L} \\
f_l &\geq -\overline{F}_l \qquad \forall l \in \mathcal{L} \\
-f_l &\geq -\overline{F}_l \qquad \forall l \in \mathcal{L}
\end{aligned} \tag{F.13}$$

has a feasible solution. Since $\tilde{p} \geq 0$ and since the injections satisfy global balance, that is

$$\sum_{s \in \mathcal{S}} (D_s - \tilde{\ell}_s) = \sum_{g \in \mathcal{G}} \tilde{p}_g, \tag{F.14}$$

we have that

$$\|D - \tilde{\ell} - \tilde{p}\|_1 \leq \|D - \tilde{\ell}\|_1 + \|\tilde{p}\|_1 = 2\|D - \tilde{\ell}\|_1 \leq 2\|D\|_1,$$

where the first inequality follows from the triangle inequality, the equality comes from (F.14) and the fact that $\tilde{\ell} \leq D$ and $0 \leq \tilde{p}$, and the last inequality again follows from $\tilde{\ell} \leq D$.

Since

$$\overline{F}_l \geq 2\sqrt{\frac{B_{\max}}{B_{\min}}}\frac{\sqrt{r(G)-1}}{2}\|D\|_1 \geq \sqrt{\frac{B_{\max}}{B_{\min}}}\frac{\sqrt{r(G)-1}}{2}\|D-\tilde{\ell}-\tilde{p}\|_1,$$

the feasibility of system (F.13) follows from Theorem 1, completing the proof. □

F.4 Proof of Proposition 2.

Let $n \in \mathbb{N}$ be given. We construct $G^n(V^n, E^n)$ as follows:

- $|V^n| = 3n$. Number the nodes from 1 to $3n$.
- For $i = 1, 2, \ldots, 3n$, define the injections as follows:

$$d_i^n = \begin{cases} -n & \text{if } i = 3n \\ 1 & \text{if } 1 \equiv i \pmod 3 \\ 0 & \text{otherwise.} \end{cases}$$

- Let

$$E^n = \left(\bigcup_{i=0}^{n-1}\{(3i+1, 3i+2), (3i+2, 3i+3), (3i+1, 3i+3)\}\right) \cup \left(\bigcup_{i=0}^{n-2}\{(3i+3, 3i+4)\}\right)$$

That is, G^n is composed of n triangles and no other cycles. As an example, the digraph for $n = 3$ is shown in Figure 3.1. By construction, for all such digraphs $G^n(V^n, E^n)$, $r(G^n) = 3$. Let $c = \frac{1}{2\sqrt{3}}$. Let

$$\overline{F}_e = c\sqrt{\frac{B_{\max}}{B_{\min}}}\frac{\sqrt{r(G^n)-1}}{2}\|d^n\|_1 = \frac{1}{2\sqrt{3}} \cdot \frac{\sqrt{3}}{2} \cdot (2n) = \frac{n}{2}$$

for all that non-cut-arcs, i.e., $e \in \{(3i+1, 3i+2), (3i+2, 3i+3), (3i+1, 3i+3) | i = 0, 1, \ldots, n-1\}$. Let $\overline{F}_e = n$ for the remaining arcs, i.e., $e \in \{(3i+3, 3i+4) | i = 0, 1, \ldots, n-2\}$.

There is a flow-polytope feasible flow on G^n given by

$$f_{(3i+1,3i+2)} = f_{(3i+2,3i+3)} = f_{(3i+1,3i+3)} = \frac{i+1}{2}$$

for $i = 0, 1, \ldots, n-1$ and

$$f_{(3i+3,3i+4)} = i+1$$

for $i = 0, 1, \ldots, n-2$. However, we can show that these injections are not DCOPF-feasible. To see this, consider the triangle formed by nodes $3n-2$, $3n-1$, and $3n$. We know that we have an in-flow of n units to node $3n-2$, and that all of it must be routed to node $3n$. Without loss of generality, suppose the phase angle at node $3n$ is 0. There are exactly two paths from $3n-1$ to $3n$: the arc between them, and the two-arc path via $3n-1$. Each of these paths has capacity $n/2$, so we must use both paths at capacity. However, this is impossible, as it requires setting the phase angle at node $3n-1$ to $n/2$ and setting θ_{3n-2} to n. But that means the flow on the arc $(3n-2, 3n)$ is $n > \overline{F}_{(3n-2,3n)} = \frac{n}{2}$. □

F.5 The Attacker MIP

The attacker MIP is given by the following formulation, which combines the constraints on the attacker decisions from Section 4.2.1 with the network flow restriction from Chapter 3.

$$\max_{\substack{\delta,u,v,w,\varphi,\\ \psi,\lambda^+,\lambda^-,\gamma,\mu}} \quad -\sum_{l\in\mathcal{L}} \overline{F}_l(v_l\lambda_l^+ + v_l\lambda_l^-) - \sum_{g\in\mathcal{G}} \overline{P}_g w_g \gamma_g + \sum_{s\in\mathcal{S}} D_s\big((1-u_s)\varphi_s + \mu_s - \psi_s\big) \tag{F.15a}$$

$$\text{s.t.} \quad \text{(4.15)-(4.25), (3.3b)-(3.3i), (3.4)-(3.9)}$$

(F.15b)

We refer the reader to Chapter 3 for the proof of the correctness of the formulation. Note that while the objective contains bilinearities, they are products of binaries and bounded continuous variables, and are therefore easily linearized with the addition of auxiliary variables, similarly to formulation (3.10).

REFERENCES

Alderson, D., G. Brown, M. Carlyle, and R. K. Wood (Apr. 2011). "Solving Defender-Attacker-Defender Models for Infrastructure Defense". In: *Operations Research, Computing, and Homeland Defense, Proceedings of the 12th INFORMS Computing Society Conference*. Ed. by R. K. Wood and R. F. Dell. Hanover, MD: INFORMS, pp. 28–49.

Alguacil, N., A. Delgadillo, and J. M. Arroyo (2014). "A trilevel programming approach for electric grid defense planning". In: *Computers & Operations Research* 41, pp. 282–290.

Álvarez, R. (2004). "Interdicting electrical power grids". MA thesis. Monterey, CA: Naval Postgraduate School.

Arguello, B., E. S. Johnson, and J. L. Gearhart (2021). *A Trilevel Model for Segmentation of the Power Transmission Grid Cyber Network*. Tech. rep. 2108.10958. ArXiv.

Arroyo, J. M. (2010). "Bilevel programming applied to power system vulnerability analysis under multiple contingencies". In: *IET Generation, Transmission Distribution* 4.2, pp. 178–190.

Australian Cyber Security Centre (2020). *Implementing Network Segmentation and Segregation*. Australian Signals Directorate.

Babaeinejadsarookolaee, S. et al. (2019). *The Power Grid Library for Benchmarking AC Optimal Power Flow Algorithms*.

Bienstock, D. and A. Verma (2010). "The N-k Problem in Power Grids: New Models, Formulations, and Numerical Experiments". In: *SIAM Journal on Optimization* 20.5, pp. 2352–2380.

Blumsack, S. (May 2006). "Network Topologies and Transmission Investment Under Electric-Industry Restructuring". PhD thesis. Pittsburgh, PA: Carnegie Mellon University.

Blumsack, S., L. B. Lave, and M. Ilić (2007). "A Quantitative Analysis of the Relationship Between Congestion and Reliability in Electric Power Networks". In: *Energy Journal* 28.4, pp. 73–100.

Bynum, M. L., G. A. Hackebeil, W. E. Hart, C. D. Laird, B. L. Nicholson, J. D. Siirola, J.-P. Watson, and D. L. Woodruff (2021). *Pyomo — optimization modeling in Python*. Third. Vol. 67. Springer Optimization and its Applications. New York: Springer.

Carpentier, J. (1962). "Contributions to the economic dispatch problem." In: *Bulletin Society Francaise Electriciens* 8.3, pp. 431–447.

Castillo, A., B. Arguello, G. Cruz, and L. Swiler (2019). "Cyber-Physical Emulation and Optimization of Worst-Case Cyber Attacks on the Power Grid". In: *2019 Resilience Week (RWS)*. Vol. 1, pp. 14–18.

Chestnut, S. R. and R. Zenklusen (July 2017). "Hardness and Approximation for Network Flow Interdiction". In: *Networks* 69.4, pp. 378–387.

Coffrin, C., H. L. Hijazi, K. Lehmann, and P. Van Hentenryck (Aug. 2014). "Primal and dual bounds for Optimal Transmission Switching". In: *2014 Power Systems Computation Conference*, pp. 1–8.

Coffrin, C., R. Bent, B. Tasseff, K. Sundar, and S. Backhaus (Mar. 2019). "Relaxations of AC Maximal Load Delivery for Severe Contingency Analysis". In: *IEEE Transactions on Power Systems* 34.2, pp. 1450–1458.

Delgadillo, A., J. M. Arroyo, and N. Alguacil (2010). "Analysis of Electric Grid Interdiction With Line Switching". In: *IEEE Transactions on Power Systems* 25.2, pp. 633–641.

Ding, T., L. Yao, and F. Li (2018). "A multi-uncertainty-set based two-stage robust optimization to defender–attacker–defender model for power system protection". In: *Reliability Engineering & System Safety* 169, pp. 179–186.

Executive Office of the President Council of Economic Advisers (2013). *Economic Benefits of Increasing Electric Grid Resilience to Weather Outages*.

Fang, Y. and E. Zio (2017). "Optimizing the resilience of interdependent infrastructure systems against intentional attacks". In: *2017 2nd International Conference on System Reliability and Safety (ICSRS)*, pp. 62–67.

Fischetti, M., I. Ljubić, M. Monaci, and M. Sinnl (2017). "A New General-Purpose Algorithm for Mixed-Integer Bilevel Linear Programs". In: *Operations Research* 65.6, pp. 1615–1637.

Fisher, E. B., R. P. O'Neill, and M. C. Ferris (Aug. 2008). "Optimal Transmission Switching". In: *IEEE Transactions on Power Systems* 23.3, pp. 1346–1355.

Fliscounakis, S., P. Panciatici, F. Capitanescu, and L. Wehenkel (Nov. 2013). "Contingency Ranking With Respect to Overloads in Very Large Power Systems Taking Into Account Uncertainty, Preventive, and Corrective Actions". In: *IEEE Transactions on Power Systems* 28.4, pp. 4909–4917.

Fuller, J. D., R. Ramasra, and A. Cha (Aug. 2012). "Fast Heuristics for Transmission-Line Switching". In: *IEEE Transactions on Power Systems* 27.3, pp. 1377–1386.

Garifi, K., E. S. Johnson, B. Arguello, and B. J. Pierre (2021). "Transmission Grid Resiliency Investment Optimization Model with SOCP Recovery Planning". In: *IEEE Transactions on Power Systems (Early Access)*.

Genge, B., P. Haller, and I. Kiss (2017). "Cyber-Security-Aware Network Design of Industrial Control Systems". In: *IEEE Systems Journal* 11.3, pp. 1373–1384.

Ghorbani-Renani, N., A. D. González, and K. Barker (2021). "A decomposition approach for solving tri-level defender-attacker-defender problems". In: *Computers & Industrial Engineering* 153, p. 107085.

Ghorbani-Renani, N., A. D. González, K. Barker, and N. Morshedlou (2020). "Protection-interdiction-restoration: Tri-level optimization for enhancing interdependent network resilience". In: *Reliability Engineering & System Safety* 199, p. 106907.

Glenn, C., D. Sterbentz, and A. Wright (June 2017). *Cyber Threat and Vulnerability Analysis of the U.S. Electric Sector*. Tech. rep. Idaho National Laboratory.

Gurobi Optimization, LLC (2020). *Gurobi Optimizer Reference Manual*.

Hart, W. E., J.-P. Watson, and D. L. Woodruff (2011). "Pyomo: modeling and solving mathematical programs in Python". In: *Mathematical Programming Computation* 3.3, pp. 219–260.

Hedman, K. W., R. P. O'Neill, E. B. Fisher, and S. S. Oren (Feb. 2011). "Smart Flexible Just-in-Time Transmission and Flowgate Bidding". In: *IEEE Transactions on Power Systems* 26.1, pp. 93–102.

Heller, I. and C. B. Tompkins (1956). "An extension of a theorem of Dantzig's". In: *Annals of Mathematics Studies* 38, pp. 247–254.

Horowitz, A. (Apr. 2021). *How We're Moving to Net-Zero by 2050*. (accessed: 10.11.2021.

Hua, B., R. Baldick, and R. K. Wood (2019). *Interdiction of a Mixed-Integer Linear System*. Preprint 2019-01-7013. Optimization Online.

Israeli, E. and R. K. Wood (2002). "Shortest-path network interdiction". In: *Networks* 40.2, pp. 97–111. eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1002/net.10039.

Jabarnejad, M. (2018). "Approximate optimal transmission switching". In: *Electric Power Systems Research* 161, pp. 1–7.

Jeroslow, R. G. (1985). "The polynomial hierarchy and a simple model for competitive analysis". In: *Mathematical Programming* 32, pp. 146–164.

Kleinert, T., M. Labbé, F. Plein, and M. Schmidt (2020). "Technical Note—There's No Free Lunch: On the Hardness of Choosing a Correct Big-M in Bilevel Optimization". In: *Operations Research* 68.6, pp. 1716–1721.

Kocuk, B., S. S. Dey, and X. A. Sun (2016). "Strong SOCP Relaxations for the Optimal Power Flow Problem". In: *Operations Research* 64.6, pp. 1177–1196. eprint: https://doi.org/10.1287/opre.2016.1489.

Kocuk, B., H. Jeon, S. S. Dey, J. Linderoth, J. Luedtke, and X. A. Sun (2016). "A Cycle-Based Formulation and Valid Inequalities for DC Power Transmission Problems with Switching". In: *Operations Research* 64.4, pp. 922–938.

Lai, K., M. Illindala, and K. Subramaniam (2019). "A tri-level optimization model to mitigate coordinated attacks on electric power systems in a cyber-physical environment". In: *Applied Energy* 235, pp. 204–218.

Lehmann, K., A. Grastien, and P. Van Hentenryck (Nov. 2014). "The Complexity of DC-Switching Problems". In.

Li, Z., M. Shahidehpour, A. Alabdulwahab, and A. Abusorrah (2016). "Bilevel Model for Analyzing Coordinated Cyber-Physical Attacks on Power Systems". In: *IEEE Transactions on Smart Grid* 7.5, pp. 2260–2272.

Liu, C., J. Wang, and J. Ostrowski (Nov. 2012). "Heuristic Prescreening Switchable Branches in Optimal Transmission Switching". In: *IEEE Transactions on Power Systems* 27.4, pp. 2289–2290.

Lozano, L. and J. C. Smith (2017a). "A Backward Sampling Framework for Interdiction Problems with Fortification". In: *INFORMS Journal on Computing* 29.1, pp. 123–139.

Lozano, L. and J. C. Smith (2017b). "A Value-Function-Based Exact Approach for the Bilevel Mixed-Integer Programming Problem". In: *Operations Research* 65.3, pp. 768–786.

Motto, A. L., J. M. Arroyo, and F. D. Galiana (2005). "A mixed-integer LP procedure for the analysis of electric grid security under disruptive threat". In: *IEEE Transactions on Power Systems* 20.3, pp. 1357–1365.

Nagarajan, H., R. Bent, P. V. Hentenryck, S. Backhaus, and E. Yamangil (Mar. 2017). "Resilient Transmission Grid Design: AC Relaxation vs. DC approximation". In.

O'Neill, R., T. Dautel, and E. Krall (2011). *Recent ISO Software Enhancements and Future Software and Modeling Plans*. Tech. rep. Federal Energy Regulatory Commission (FERC).

Oster, M., S. Chatterjee, F. Pan, C. Bakker, A. Bhattacharya, and C. Perkins (2020). "Power system resilience through defender-attacker-defender models with uncertainty: an overview". In: *2020 Resilience Week (RWS)*, pp. 11–17.

Ouyang, M. (2017). "A mathematical framework to optimize resilience of interdependent critical infrastructure systems under spatially localized attacks". In: *European Journal of Operational Research* 262.3, pp. 1072–1084.

Ouyang, M. and Y. Fang (2017). "A Mathematical Framework to Optimize Critical Infrastructure Resilience against Intentional Attacks". In: *Computer-Aided Civil and Infrastructure Engineering* 32.11, pp. 909–929.

Ouyang, M., F. Tao, S. Huang, M. Xu, and C. Zhang (2018). "Vulnerability Mitigation of Multiple Spatially Localized Attacks on Critical Infrastructure Systems". In: *Computer-Aided Civil and Infrastructure Engineering* 33.7, pp. 585–601.

Ouyang, M., M. Xu, C. Zhang, and S. Huang (2017). "Mitigating electric power system vulnerability to worst-case spatially localized attacks". In: *Reliability Engineering & System Safety* 165, pp. 144–154.

Pillitteri, V. Y. and T. L. Brewer (2014). *Guidelines for Smart Grid Cybersecurity*. Interagency/Internal Report (NISTIR)-7628, Revision 1. National Institute of Standards (NIST).

Pineda, S. and J. M. Morales (2018). "Solving Linear Bilevel Problems Using Big-Ms: Not All That Glitters Is Gold". In: *IEEE Transactions on Power Systems* 34, pp. 2469–2471.

Roald, L. A. and D. K. Molzahn (2019). "Implied Constraint Satisfaction in Power System optimization: The Impacts of Load Variations". In: *2019 57th Annual Allerton Conference on Communication, Control, and Computing (Allerton)*, pp. 308–315.

Ruiz, P. A., J. M. Foster, A. Rudkevich, and M. C. Caramanis (Aug. 2012). "Tractable Transmission Topology Control Using Sensitivity Analysis". In: *IEEE Transactions on Power Systems* 27.3, pp. 1550–1559.

Ruiz, P. A., E. Goldis, A. M. Rudkevich, M. C. Caramanis, C. R. Philbrick, and J. M. Foster (Mar. 2017). "Security-Constrained Transmission Topology Control MILP Formulation Using Sensitivity Factors". In: *IEEE Transactions on Power Systems* 32.2, pp. 1597–1605.

Ruiz, P. A., A. Rudkevich, M. C. Caramanis, E. Goldis, E. Ntakou, and C. R. Philbrick (Oct. 2012). "Reduced MIP formulation for transmission topology control". In: *2012 50th Annual Allerton Conference on Communication, Control, and Computing (Allerton)*, pp. 1073–1079.

Salmeron, J., K. Wood, and R. Baldick (2004a). "Analysis of electric grid security under terrorist threat". In: *IEEE Transactions on Power Systems* 19.2, pp. 905–912.

Salmeron, J., K. Wood, and R. Baldick (2009). "Worst-Case Interdiction Analysis of Large-Scale Electric Power Grids". In: *IEEE Transactions on Power Systems* 24.1, pp. 96–104.

Salmeron, J., K. Wood, and R. Baldick (Mar. 2004b). *Optimizing Electric Grid Design Under Asymmetric Threat (II)*. Tech. rep. Naval Postgraduate School.

Sayyadipour, S., G. R. Yousefi, and M. A. Latify (2016). "Mid-term vulnerability analysis of power systems under intentional attacks". In: *IET Generation, Transmission Distribution* 10.15, pp. 3745–3755.

Scaparra, M. P. and R. L. Church (June 2008). "A Bilevel Mixed-Integer Program for Critical Infrastructure Protection Planning". In: *Computers & Operations Research* 35.6, pp. 1905–1923.

Smith, J. C. and Y. Song (2020). "A survey of network interdiction models and algorithms". In: *European Journal of Operational Research* 283.3, pp. 797–811.

Stouffer, K., V. Pillitteri, M. Suzanne Lightman, and A. A. Hahn (2016). *Guide to Industrial Control Systems (ICS) Security*. Special Publication 800-82, Revision 2. National Institute of Standards (NIST).

Sun, C.-C., A. Hahn, and C.-C. Liu (2018). "Cyber security of a power grid: State-of-the-art". In: *International Journal of Electrical Power & Energy Systems* 99, pp. 45–56.

Sundar, K., C. Coffrin, H. Nagarajan, and R. Bent (2018). "Probabilistic N-k failure-identification for power systems". In: *Networks* 71.3, pp. 302–321. eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1002/net.21806.

Sundar, K., S. Misra, R. Bent, and F. Pan (2021). "Credible Interdiction for Transmission Systems". In: *IEEE Transactions on Control of Network Systems (Early Access)*, pp. 1–11.

Sundar, K., S. Misra, R. Bent, and F. Pan (Apr. 2019). "Spatial and Topological Interdiction for Transmission Systems". In.

Tahernejad, S., T. K. Ralphs, and S. T. DeNegre (2020). "A branch-and-cut algorithm for mixed integer bilevel linear optimization problems and its implementation". In: *Mathematical Programming Computation* 12.4, pp. 529–568.

Tang, Y., J.-P. P. Richard, and J. C. Smith (Oct. 2016). "A class of algorithms for mixed-integer bilevel min–max optimization". In: *Journal of Global Optimization* 66.2, pp. 225–262.

Ton, D. T. and W. P. Wang (May 2015). "A More Resilient Grid: The U.S. Department of Energy Joins with Stakeholders in an R&D Plan". In: *IEEE Power and Energy Magazine* 13.3, pp. 26–34.

U.S. Energy Information Administration (Sept. 2021). *September 2021 Monthly Energy Review*.

US Department of Homeland Security, Industrial Control Systems Cyber Emergency Response Team (2016). *Recommended Practice: Improving Industrial Control System Cybersecurity with Defense-in-Depth Strategies*.

Wang, Y. and R. Baldick (2014). "Interdiction Analysis of Electric Grids Combining Cascading Outage and Medium-Term Impacts". In: *IEEE Transactions on Power Systems* 29.5, pp. 2160–2168.

Wood, R. K. (2011). "Bilevel Network Interdiction Models: Formulations and Solutions". In: *Wiley Encyclopedia of Operations Research and Management Science*. Ed. by J. J. Cochran, L. A. Cox Jr., P. Keskinocak, J. P. Kharoufeh, and J. C. Smith. New York: John Wiley & Sons.

Wu, X. and A. J. Conejo (2017). "An Efficient Tri-Level Optimization Model for Electric Grid Defense Planning". In: *IEEE Transactions on Power Systems* 32.4, pp. 2984–2994.

Xavier, A. S., F. Qiu, and S. Ahmed (2020). "Learning to Solve Large-Scale Security-Constrained Unit Commitment Problems". In: *INFORMS Journal on Computing* Articles in Advance, pp. 1–18.

Yang, Z. and S. Oren (June 2019). "Line Selection and Algorithm Selection for Transmission Switching by Machine Learning Methods". In: *2019 IEEE Milan PowerTech*, pp. 1–6.

Yao, Y., T. Edmunds, D. Papageorgiou, and R. Alvarez (2007). "Trilevel Optimization in Power Network Defense". In: *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)* 37.4, pp. 712–718.

Yuan, W., L. Zhao, and B. Zeng (2014). "Optimal power grid protection through a defender–attacker–defender model". In: *Reliability Engineering & System Safety* 121, pp. 83–89.

Yue, D., J. Gao, B. Zeng, and F. You (2019). "A projection-based reformulation and decomposition algorithm for global optimization of a class of mixed integer bilevel linear programs". In: *Journal of Global Optimization* 73, pp. 27–57.

Zeng, B. and L. Zhao (2013). "Solving two-stage robust optimization problems using a column-and-constraint generation method". In: *Operations Research Letters* 41.5, pp. 457–461.

Zhao, L. and B. Zeng (2013). "Vulnerability Analysis of Power Grids With Line Switching". In: *IEEE Transactions on Power Systems* 28.3, pp. 2727–2736.