NAVAL POSTGRADUATE SCHOOL

MONTEREY, CALIFORNIA

THESIS

DEFENDING CRITICAL INFRASTRUCTURE AGAINST DELIBERATE THREATS AND NON-DELIBERATE HAZARDS

by

Jason D. Ross

September 2014

Thesis Advisor: David L. Alderson
Second Reader: W. Matthew Carlyle

Approved for public release; distribution is unlimited

THIS PAGE INTENTIONALLY LEFT BLANK

REPORT DOCUMENTATION PAGE

Form Approved OMB No. 0704–0188

1. AGENCY USE ONLY ***(Leave blank)***	**2. REPORT DATE** September 2014	**3. REPORT TYPE AND DATES COVERED** Master's Thesis

4. TITLE AND SUBTITLE
DEFENDING CRITICAL INFRASTRUCTURE AGAINST DELIBERATE THREATS AND NON-DELIBERATE HAZARDS

5. FUNDING NUMBERS

6. AUTHOR(S) Jason D. Ross

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
Naval Postgraduate School
Monterey, CA 93943–5000

8. PERFORMING ORGANIZATION REPORT NUMBER

9. SPONSORING /MONITORING AGENCY NAME(S) AND ADDRESS(ES)
N/A

10. SPONSORING/MONITORING AGENCY REPORT NUMBER

11. SUPPLEMENTARY NOTES The views expressed in this thesis are those of the author and do not reflect the official policy or position of the Department of Defense or the U.S. Government. IRB Protocol number ____N/A____.

12a. DISTRIBUTION / AVAILABILITY STATEMENT
Approved for public release; distribution is unlimited

12b. DISTRIBUTION CODE

13. ABSTRACT (maximum 200 words)

This study determines how to invest limited resources to increase the resilience of an infrastructure system against both non-deliberate and deliberate events. We propose an optimization model that seeks the best defensive investment for a weighted combination of deliberate and non-deliberate events. We formulate the general problem and conduct numerical analysis using a specific infrastructure system as a concrete example. We perform parametric analysis on the combined model in order to explore the way in which different solutions depend on the distributed weights and yield insight into the best investment decisions.

14. SUBJECT TERMS
critical infrastructure, risk, attacker-defender, optimization, deliberate threat non-deliberate hazard

15. NUMBER OF PAGES
69

16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	**18. SECURITY CLASSIFICATION OF THIS PAGE**	**19. SECURITY CLASSIFICATION OF ABSTRACT**	**20. LIMITATION OF ABSTRACT**
Unclassified	Unclassified	Unclassified	UU

NSN 7540–01–280–5500

Standard Form 298 (Rev. 2–89)
Prescribed by ANSI Std. 239–18

THIS PAGE INTENTIONALLY LEFT BLANK

Approved for public release; distribution is unlimited

DEFENDING CRITICAL INFRASTRUCTURE AGAINST DELIBERATE THREATS AND NON-DELIBERATE HAZARDS

Jason D. Ross
Lieutenant, United States Navy
B.A., Auburn University, 2008

Submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE IN OPERATIONS RESEARCH

from the

NAVAL POSTGRADUATE SCHOOL
September 2014

Author: Jason D. Ross

Approved by: David L. Alderson
Thesis Advisor

W. Matthew Carlyle
Second Reader

Robert F. Dell
Chair, Department of Operations Research

THIS PAGE INTENTIONALLY LEFT BLANK

ABSTRACT

This study determines how to invest limited resources to increase the resilience of an infrastructure system against both non-deliberate and deliberate events. We propose an optimization model that seeks the best defensive investment for a weighted combination of deliberate and non-deliberate events. We formulate the general problem and conduct numerical analysis using a specific infrastructure system as a concrete example. We perform parametric analysis on the combined model in order to explore the way in which different solutions depend on the distributed weights and yield insight into the best investment decisions.

THIS PAGE INTENTIONALLY LEFT BLANK

TABLE OF CONTENTS

THIS PAGE INTENTIONALLY LEFT BLANK

LIST OF FIGURES

THIS PAGE INTENTIONALLY LEFT BLANK

LIST OF TABLES

THIS PAGE INTENTIONALLY LEFT BLANK

LIST OF ACRONYMS AND ABBREVIATIONS

AD	Attacker-Defender
DAD	Defender-Attacker-Defender
DED	Defender-Expectation-Defender
DHS	Department of Homeland Security
DOJ	Department of Justice
ED	Expectation-Defender
NASA	National Aeronautics and Space Administration
NIPP	National Infrastructure Protection Plan
PPD	Presidential Decision Directive
PRA	probabilistic risk assessment
U.S.	United States

THIS PAGE INTENTIONALLY LEFT BLANK

EXECUTIVE SUMMARY

Owners and operators of critical infrastructure systems face a challenge when it comes to investing limited resources to maintain continuity of function in the presence of disruptive events. On the one hand, they need to protect against the most commonly occurring disruptions (e.g., failures or weather events). On the other hand, they need to protect against worst-case disruptions, as might be caused by an intelligent adversary who uses insider knowledge to deliberately interrupt system function. Often these two types of potential disruptions point to different parts of the system as "critical," and in that case this tension can lead to a dilemma in terms of what to protect.

In this study, we consider how to invest limited resources to increase the resilience of an infrastructure system against both non-deliberate and deliberate events. We represent the function of the infrastructure as a network flow problem and use an optimization model that seeks the best defensive investment for a weighted combination of deliberate and non-deliberate events. We conduct parametric analysis on the combined model in order to explore the way in which different solutions depend on the distributed weights and yield insight into the best investment decisions. From our analysis, we discover that it is possible to choose the best defense for that specific network by mapping the frontier between the worst-case and random disruptions. The results of that frontier gives us insight into (1) the minimum number of edges to protect, (2) the maximum defense for both worst-case and random disruptions, and (3) a moderate cost-effective defense for both worst-case and random disruptions

Our investigation of a small network system provides proof-of-concept that the technique we propose can be effective, but there is more work to be done. In particular, our solutions are computationally intensive even for small systems, and additional research to reduce the required computational effort is needed before the technique can be used at large scale.

THIS PAGE INTENTIONALLY LEFT BLANK

ACKNOWLEDGMENTS

I would like thank my wife, Krystal Ross, for her support during the whole education process at Naval Postgraduate School (NPS). The Operations Research department as a whole has been instrumental and effective throughout my learning journey at NPS. I must acknowledge the diligent work and support of Professor Mathew Carlyle; he is more than just a second reader—formulation and python assistance were crucial to thesis completion. I must also thank Professor David Alderson for giving me the topic and laying out the blueprint for success. These two professors are top notch and deserve all the credit for the completion of this thesis.

THIS PAGE INTENTIONALLY LEFT BLANK

I. INTRODUCTION

The United States (U.S.) has critical infrastructure systems (e.g., electric power, transportation, telecommunications, water) that need to be protected from disruptive events that are non-deliberate (e.g., natural disasters, accidents, failures) and deliberate (e.g., vandalism, sabotage, terrorism). Recently, these systems have been the targets terrorist attacks (September 11, 2001) and natural disasters (Hurricane Katrina in 2005 and Hurricane Sandy in 2012) which caused billions of dollars in damage. In response to these and other disruptive events, the U.S. is spending billions of tax payer dollars annually (e.g., $50 billion in 2014) to strengthen infrastructure (The White House 2014). Of particular interest is how to allocate these funds in a manner that increases the resilience of our critical infrastructure systems.

In the reliability engineering literature (e.g., Hwang et al. 1981 and Billinton and Allan 1992) and the risk literature (e.g., Bedford and Cook 2001), non-deliberate disruptions are commonly modeled as random events using probabilities. However, deliberate disruptions are more appropriately represented as worst-case events using game theory (e.g., Brown et al. 2006). These two perspectives often point to different parts of an infrastructure system as "critical." This leaves policy makers in a quandary about how to invest a limited defensive budget to make the infrastructure system more resilient.

The goal of this study is to determine how to invest limited resources to increase the resilience of an infrastructure against both non-deliberate and deliberate events. This information will provide decision makers with insight into which assets to protect and how much emphasis to put toward preparing for natural disasters vice a terrorist attack.

We propose an optimization model that seeks the best defensive investment for a weighted combination of deliberate and non-deliberate events. We formulate the general problem and conduct numerical analysis using a specific infrastructure system as a concrete example. We perform parametric analysis on the weights in the combined

model in order to explore the way in which different solutions depend on the distributed weights and yield insight into the best investment decisions.

II. LITERATURE REVIEW

Presidential Decision Directive 21 (PPD21), issued in February 2013, establishes national policy on critical infrastructure security and resilience (The White House 2013). Following the definition established previously in the USA PATRIOT Act (Department of Justice (DOJ) 2001), PPD21 defines critical infrastructure to include "systems and assets, whether physical or virtual, so vital to the United States that the incapacity or destruction of such systems and assets would have a debilitating impact on security, national economic security, national public health or safety, or any combination of those matters."

PPD21 specifically states "It is the policy of the United States to strengthen the security and resilience of its critical infrastructure… to take proactive steps to manage risk and strengthen the security and resilience of the Nation's critical infrastructure, considering all hazards that could have a debilitating impact on national security, economic stability, public health and safety, or any combination thereof. These efforts shall seek to reduce vulnerabilities, minimize consequences, identify and disrupt threats, and hasten response and recovery efforts related to critical infrastructure." In its definition of "all hazards," PPD21 refers to "a threat or an incident, natural or manmade [that]… includes natural disasters, cyber incidents, industrial accidents, pandemics, acts of terrorism, sabotage, and destructive criminal activity targeting critical infrastructure."

In response to PPD21, the Department of Homeland Security (DHS) issued an updated National Infrastructure Protection Plan (NIPP) in December 2013. The 2013 edition of the NIPP "continues to focus on risk management as the foundation of critical infrastructure security and resilience" (p. 4).

The application of risk analysis to critical infrastructure systems follows a long history in the application of probabilistic risk assessment (PRA) to complex engineering systems (see Bedford and Cook 2001 for an introduction, or National Aeronautics and Space Administration (NASA) 2011 for a recent application of PRA to space mission planning). Following the attacks of 9/11, much of the work on risk to critical

infrastructure has focused on the application of PRA to terrorism risk (e.g., Pate-Cornell and Guikema 2002, Garrick et al. 2004, Parnell, et al. 2005, Willis 2007). The Department of Homeland Security has promoted the use of probabilistic techniques when assessing the risk critical infrastructure systems (DHS 2013).

A different line of research has focused on the ability of an intelligent adversary to deliberately disrupt system function. Specifically, there is now considerable work on the use of system interdiction models for understanding homeland security problems and the risk to critical infrastructure systems—these models are known as Attacker-Defender (AD) and Defender-Attacker-Defender (DAD) problems (Brown et al., 2005, Brown et al., 2006, Alderson et al., 2014a, b). Applications of this model have been used in a variety of systems: fuel (Ileto 2011), cargo (De La Cruz 2011), the electric grid (Salmeron et al. 2004, 2009), and military networks (Burton 2013, Long 2013).

In this thesis, we propose a model to defend our infrastructure against both non-deliberate hazards and deliberate threats. We use a linear combination of the two techniques previously described in order to map the efficient frontier of the tradeoffs between the consequences of a worst-case attack and expected consequence over a set of random events that will give decision makers the ability to choose the type of protection needed for his or her particular network.

Previous work by Bier (2007), Zhuang and Bier (2007), and Hausken et al. (2009) has looked at the challenge of choosing what to protect, particularly when facing “all hazards” to include deliberate and non-deliberate events. However, these studies have abstracted the consequence of the event to a simple known function. As demonstrated by Alderson et al. (2013), in general it is difficult to approximate the consequences of losing one or more infrastructure components, or to pick the most vital components in a network system. An explicit objective of this research is to connect the operational models of infrastructure function and consequence assessment with the need to identify protective measures that work for both deliberate and non-deliberate disruptive events.

III. MODEL DEVELOPMENT

In this chapter, we present a sequence of models that we use to perform our analysis. Throughout, we follow the concepts and notation in Alderson et al. (2014a). A specific implementation of these models follows in Chapter IV.

A. NORMAL OPERATION

In this section, we provide an overview of the sequence of models to follow. Our starting point is an Operator Model of the form

$$\min_{y \in Y(\hat{w})} f(\hat{w}, \hat{x}, y), \tag{1}$$

where $\hat{w}$ denotes the fixed design of the system, $\hat{x}$ denotes the operational setting (e.g., the status of each component of the system), and $Y(\hat{w})$ denotes the set of feasible activities available to the system operator, given design $\hat{w}$.

The Operator Model represents the decision making of an entity, called the operator, who chooses the activities in the system. The Operator Model is prescriptive; its solution $y^* = \arg\min_{y \in Y(\hat{w})} f(\hat{w}, \hat{x}, y)$ represents the activities in the system that yield the minimum operating cost.

The Operator Model (1) is a mathematical program that can be solved using various techniques (see Alderson et al. 2014a for a detailed discussion).

B. WORST-CASE DISRUPTIONS

We are interested in understanding how well the system performs in the presence of worst-case disruptions and/or most likely disruptions. For a fixed design $\hat{w}$, we define

$$P(\hat{w}) = \max_{x \in X} \min_{y \in Y(\hat{w})} f(\hat{w}, \hat{x}, y), \tag{2}$$

where X is the set of feasible disruptive events, and $x^* = \arg\max_{x \in X} \min_{y \in Y(\hat{w})} f(\hat{w}, \hat{x}, y)$ is a worst-case disruptive event. Here, $P(\hat{w})$ represents the performance of the system in the presence of the worst-case disruptive event. We refer to (2) as the Attacker Model; it builds on the previous formulation but has additional elements.

In general, *P(* $\hat{w}$*)* can be evaluated three ways:

1. by exhaustive enumeration of $x \in X$,

2. by taking the dual of the inner Operator Model to transform the max-min of the Attacker Model into a max-max problem (e.g., Brown et al., 2006), and

3. using Bender's decomposition (e.g., Benders, 1962).

Given a set of possible designs W , our goal is to find a design $w \in W$ that yields "good" performance in the presence of disruptive events.

If we are only concerned with defending against worst-case disruptions, then we solve a problem of the form: $\min_{w \in W} P(w)$. This is simply a deterministic Defender-Attacker-Defender (DAD) problem. Let $W_D^* \subset W$ denote the set of optimal designs that minimize *P(w)*. That is, $w^* \in W_D^* \Leftrightarrow w^* = \arg\min_{w \in W} P(w)$.

In principle, solving the Defender Model requires nothing more than enumerating every possible combination of defense and attack, then solving the corresponding Operator Model for each, and then finding the one that yields the lowest cost. This is the first half of our combined model which will determine the effect of the worst case. The second half of the model is the probabilistic model that optimizes defense against hazards.

C. PROBABILISTIC DISRUPTION (MOST LIKELY)

For a fixed design $\hat{w}$, define

$$Q(\hat{w}) = E_{\tilde{x}} \min_{y \in Y(\hat{w})} f(\hat{w}, \tilde{x}, y), \tag{3}$$

where $\tilde{x}$ is a random variable having a specified distribution, and $E_{\tilde{x}}$ denotes the expectation relative to $\tilde{x}$. Here, *Q(* $\hat{w}$*)* represents the average performance of the system in the presence of a random disruptive event.

In general, *Q(* $\hat{w}$*)* can be evaluated two ways:

1. via Monte Carlo sampling on $\tilde{x}$, and

2. using exhaustive enumeration if $\tilde{x}$ is discrete to compute an exact expectation.

If we are only concerned with defending against random disruptive events, then we solve a problem of the form: $\min_{w \in W} Q(w)$. This is simply a stochastic optimization problem. Let $W_s^* \in W$ denote the set of optimal designs that minimize *Q(w)*. That is, $w^* \in W_s^* \Leftrightarrow w^* = \arg\min_{w \in W} Q(w)$.

D. DEFENDING AGAINST WORST-CASE AND MOST LIKELY

In practice, we are concerned with both random disruptive events and worst-case disruptive events. To determine a good design, we must take into considerations the following two possibilities.

1. Observe that if $W_D^* \cap W_S^* \neq \varnothing$, then we have an optimal solution to both problems and there is no issue. We simply choose $w^* \in (W_D^* \cap W_S^*)$.

2. However, in general we expect $W_D^* \cap W_S^* = \varnothing$. More strongly, our experience is often that the worst-case disruptive event is very different from the average case disruptive event. This creates a dilemma in terms of where to invest defensive resources to make the system more resilient.

We propose to study the combined problem as

$$Z^*(\lambda) = \min_{w \in W} [\lambda P(w) + (1-\lambda) Q(w)] \quad \text{where } 0 \leq \lambda \leq 1. \tag{4}$$

Note that when $\lambda = 0$, Eq.(4) corresponds to the stochastic average-case. In contrast, when $\lambda = 1$, Eq.(4) corresponds to the deterministic worst-case.

THIS PAGE INTENTIONALLY LEFT BLANK

IV. CASE STUDY

In this thesis, we use a small numerical example to illustrate our technique. Following Alderson et al. (2014b), Figure 1 displays a notional (fuel distribution) infrastructure system made of nodes and links. The system is a single commodity transshipment model, designed to move fuel from supply locations to demand locations.

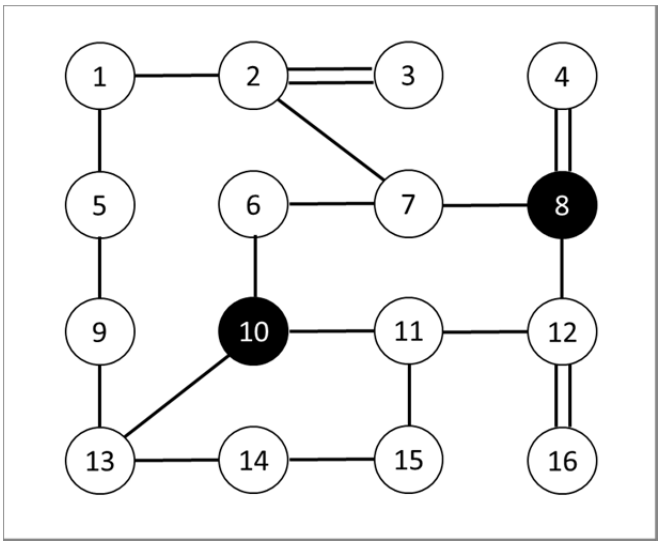

Figure 1. Notional fuel distribution network (from Alderson et al. 2014b).

In Figure 1, the white circles represent locations with demand = 1 and the black circles represent locations with supply = 10. All links are bidirectional and have per unit flow usage cost = 1. Each link has capacity = 15 and the penalty for unsatisfied demand per node = 10. Usage penalty for an interdicted arc = 20 per unit of flow (e.g., hiring a truck to drive the unit of flow to bypass the link). The objective is to minimize the sum of the flow cost for fuel deliveries and any penalties from unsatisfied demands.

As shown in Figure 1, Nodes 3, 4, and 16 have two (parallel, redundant) connections to the rest of the network. This network has been built to be *N*-1 reliable, meaning that the loss of any single link does not disconnect any node.

A. OPERATOR MODEL

We formulate the Operator Model for this system as follows.

Indices and Sets

$n \in N$	nodes (alias i,j)
$[i,j] \in E$	undirected edge between nodes i and j, $i<j$
$(i,j) \in A$	directed arc from node i to node j
	$[i,j] \in E \Leftrightarrow (i<j) \wedge ((i,j) \in A \wedge (j,i) \in A))$
$d \in D$	defenses, $D=\{0,1\}$: 0 represents undefended, 1 represents defended (and invulnerable)

Data [units]

c_{ij}	per unit cost of traversing arc $(i,j) \in A$ [dollars/barrel]
u_{ij}	upper bound on total (undirected) flow on edge $[i,j] \in E$ [barrels]
$\hat{x}_{ij}$	1 if edge $[i,j] \in E$ damaged, 0 otherwise [binary]
q_{ij}^{d}	per unit penalty cost of traversing arc $(i,j) \in A$ if damaged [dollars/barrel]
b_n	fuel supply at node $n \in N$ [barrels] (-demand for $b_n <0$)
v_n	per unit penalty cost for demand shortfall $n \in N$ [dollars/barrel]
$\hat{w}_{ij}^{d}$	1 if defense option d is implemented on edge $[i,j] \in E$, 0 otherwise [binary]

Decision Variables [units]

Y_{ij}^{d} flow on arc $(i,j) \in A$ for defense d [barrels]

S_n fuel shortfall at node $n \in N$ [barrels]

Formulation:

$$\min_{Y,S} \quad \sum_{d\in D}\sum_{[i,j]\in E}\left[\left(c_{ij}+q_{ij}^{d}\hat{x}_{ij}\right)Y_{ij}^{d}+\left(c_{ji}+q_{ji}^{d}\hat{x}_{ij}\right)Y_{ji}^{d}\right]+\sum_{n\in N}v_n S_n \qquad \text{(D0)}$$

$$s.t. \quad \sum_{d\in D}\sum_{j:(n,j)\in A}Y_{nj}^{d}-\sum_{d\in D}\sum_{i:(i,n)\in A}Y_{in}^{d}-S_n \le b_n \qquad \forall n\in N \qquad [\pi_n] \qquad \text{(D1)}$$

$$Y_{ij}^{d}+Y_{ji}^{d} \le u_{ij}\hat{w}_{ij}^{d} \qquad \forall [i,j]\in A, d\in D \qquad \left[\mu_{ij}^{d}\right] \qquad \text{(D2)}$$

$$Y_{ij}^{d} \ge 0 \qquad \forall [i,j]\in A, d\in D \qquad \text{(D3)}$$

$$S_n \ge 0 \qquad \forall n\in N \qquad \text{(D4)}$$

Discussion

The objective function (D0) combines the total flow cost and the total penalty cost. Constraints (D1) enforce balance of flow at each node. Stipulations (D2) and (D3) ensure bounds on decision variables. This formulation implements cost-based interdiction—that is, damage to an arc makes it extremely expensive but not infeasible—which makes the problem easier to solve computationally. In the above example, we have b_8=b_{10}=10 and b_n=-1 otherwise. In addition, we set c_{ij}=1, $q_{ij}^{1}=0, \forall (i,j)\in A$ $u_{ij}=100, \forall (i,j)\in A$, $q_{ij}^{0}=10, \forall (i,j)\in A$, u_{ij}=15 for all $[i,j] \in E$, and v_n =10 for all $n \in N$. There are no arcs defended in the base case; therefore, $\hat{w}_{ij}^{1}$ =0 and $\hat{w}_{ij}^{0}$ =1 for all $[i,j] \in$ E.

Solving this problem, we obtain a minimum-cost solution consisting of flows that meet all the demands (see Figure 2), with the flows labeled in blue. Our findings are consistent to the base solution findings in Alderson et al. (2014b). We are concerned with the ability of the system to perform in the presence of either random (probabilistic) or deliberate (worst-case) disruptive events.

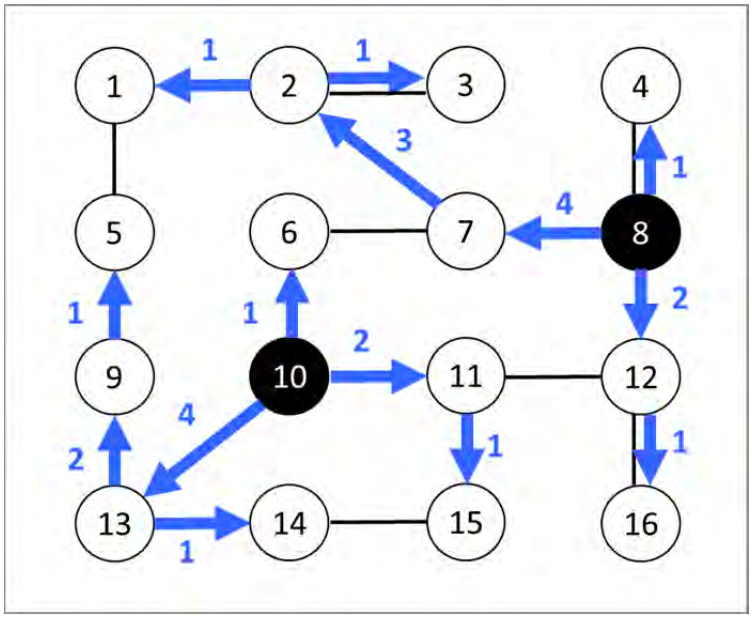

Figure 2. Operator Model base solution

B. ATTACKER MODEL

For this notional fuel system, we model the attacker's problem as follows.

Additional Data[units]

r_{ij}	"cost" to break edge $[i,j] \in E$ [cardinality]
attack_budget	budget constraint on the number of simultaneous attacks [cardinality]

Additional Decision Variables [units]

X_{ij}	1 if attacker breaks edge $[i,j] \in E$, 0 otherwise [binary]

Formulation

$$P(\hat{w}) = \max_{X} \min_{Y,S} \sum_{d \in D} \sum_{[i,j] \in E} \left[(c_{ij} + q_{ij}^{d} X_{ij}) Y_{ij}^{d} + (c_{ji} + q_{ji}^{d} X_{ij}) Y_{ji}^{d} \right] + \sum_{n \in N} v_n S_n \qquad \text{(AD0)}$$

$$\text{s.t.} \quad \text{(D1), (D2), (D3)(D4)}$$

$$\sum_{[i,j] \in E} r_{ij} X_{ij} \leq attack\_budget \qquad \text{(AD1)}$$

$$X_{ij} \in \{0,1\} \qquad \forall [i,j] \in E \qquad \text{(AD2)}$$

The objective function (AD0) is the same as that for the Operator Model (D0), except that parameters $\hat{x}_{ij}$ have been replaced by decision variables X_{ij}. Constraint (AD1) limits the number of simultaneous attacks, and the cost to attack each arc can be different. Stimpulations (AD2) require that attacks are binary. We note that $q_{ij} = 0$ implies that arc (i,j) is effectively invulnerable, because attacking it does not increase the flow cost for the operator.

In the above example, we model parallel arcs as costing twice as much to attack. That is, we have $r_{2,3} = r_{4,8} = r_{12,16} = 2$ and all other $r_{ij} = 1$.

The solution for the Attacker Model depends on the attacker budget (the number of attacks that the attacker can afford). The attacker seeks to maximize the cost to the operator. In Figure 3, an attack on edge [10,13] represents the worst case attack to the operator when the attacker can afford to target only a single edge. The operator reroutes his flow as shown in blue and the overall operator cost increases from 25 to 33. An attacker having the ability to target a large number of edges could render our network practically useless as shown in Figure 4. Here, the attacker has the budget to target five edges and focuses on the arcs around the supply nodes to cripple flow to the entire network with exception to node 4. The attacker picks the best place to attack depending on how many attacks allotted by the attacking budget. These results are also consistent with Alderson et al. (2014b).

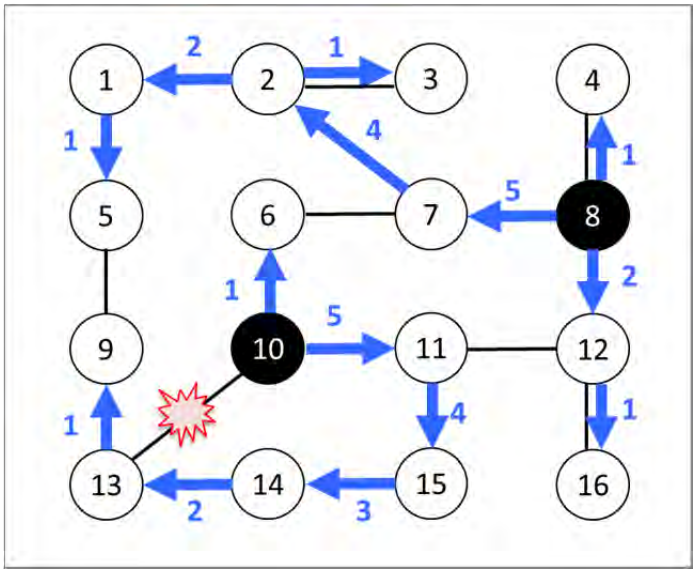

Figure 3. Operator Model solution with 1 attack

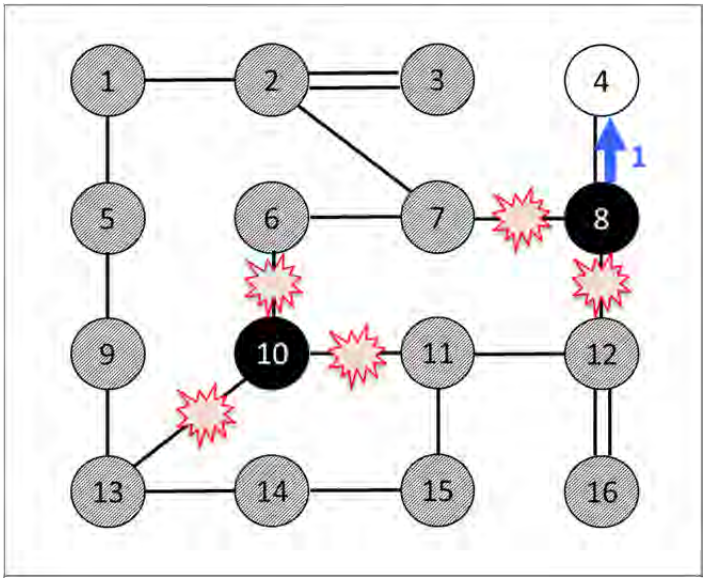

Figure 4. Operator Model solution with 5 attacks

C. DEFENDING AGAINST A WORST-CASE ATTACK

In this thesis, we restrict defensive actions to the "hardening" of edges, such that targeting an edge does not increase its usage cost. We formulate the following model.

Additional Data[units]

defense_budget budget constraint on the number of defenses [cardinality]

Additional Decision Variables [units]

W_{ij}^{d} 1 if defense option *d* is implemented on edge *[i,j]* $\in E$,

0 otherwise [binary]

Formulation

$$\min_{W}\max_{X}\min_{Y,S}\sum_{d\in D}\sum_{[i,j]\in E}\left[(c_{ij}+q_{ij}^{d}X_{ij})Y_{ij}^{d}+(c_{ji}+q_{ji}^{d}X_{ij})Y_{ji}^{d}\right]+\sum_{n\in N}v_{n}S_{n} \qquad (DAD0)$$

s.t. $(\mathrm{D1})(\mathrm{AD1})(\mathrm{AD2})(\mathrm{D3})(\mathrm{D4})$

$$Y_{ij}^{d}+Y_{ji}^{d}\le u_{ij}W_{ij}^{d} \qquad \forall[i,j]\in E, d\in D \qquad (DAD1)$$

$$\sum_{[i,j]\in E}\mathrm{W}_{ij}^{1}\le defense\_budget \qquad (DAD2)$$

$$\sum_{d}W_{ij}^{d}=1 \qquad \forall[i,j]\in E \qquad (DAD3)$$

$$W_{ij}^{d}\in\{0,1\} \qquad \forall[i,j]\in E, d\in D \qquad (DAD4)$$

The objective (DAD0) includes the cost of flow over existing (and possibly damaged or protected) edges and penalties for unmet demand. Contraints (DAD1) ensures that flow on edge [i,j] does not exceed flow capacity u_{ij}. The constraint (DAD2) requires that the cost of all defenses fall within the existing defense budget using the binary variable W_{ij}^{d}. Constraints (DAD3) ensure that flow on each edge [*i,j*] is on either the protected or unprotected edge and not both. Here, the pair W_{ij}^{1}=1 and W_{ij}^{0}=0 denotes a defended edge and vice versa for undefended. Stipulations (DAD4) require binary defense decisions.

In order to be consistent with Alderson et al. (2014b) we will give examples of defense with an attacker budget of three attacks as shown in Figure 5. For the purposes of this study we only focus on hardening edges.

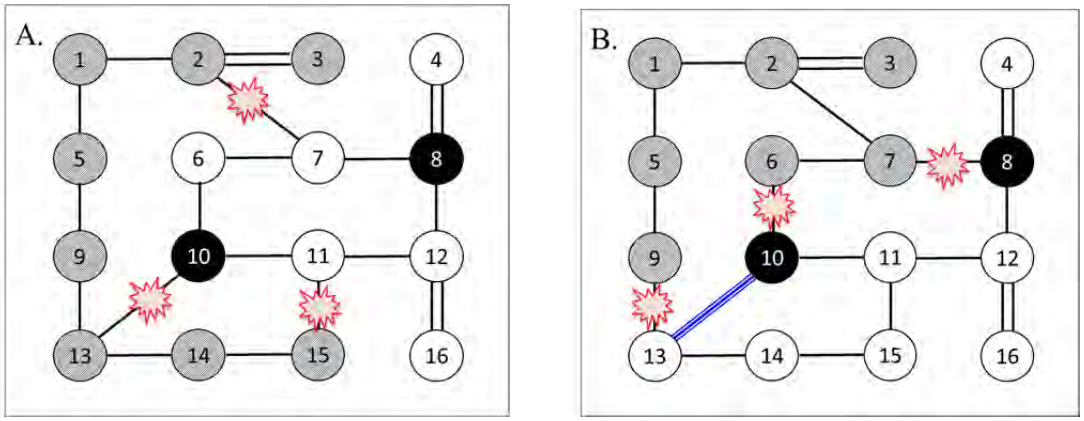

Figure 5. **A.** AD Model: worst-case attack involving three edges. In the absence of any defenses, edges [2,7], [10,13], and [11,15] are targeted. **B.** DAD Model: the corresponding worst-case attack with one edge protected. Edge [10,13] is defended, and edges [6,10], [7,8], and [9,13] are targeted.

The defender model protects edge [10, 13], note that it is the same arc the attacker chose as the one arc attack in Figure 3. The attacker picks another set of arcs to attack in order to maximize cost for the operator that does not include the defended arc. With edge [10,13] defended and in the presence of three targeted edges, the operator is able to satisfy demand at seven nodes, instead of only six with no arcs defended.

The defender can limit the attacker by strengthening more arcs at shown in Figure 6. However, we cannot defend all arcs because of budget constraints. Therefore, decisions about what arcs to protect must be made within those constraints to minimize cost to the operator.

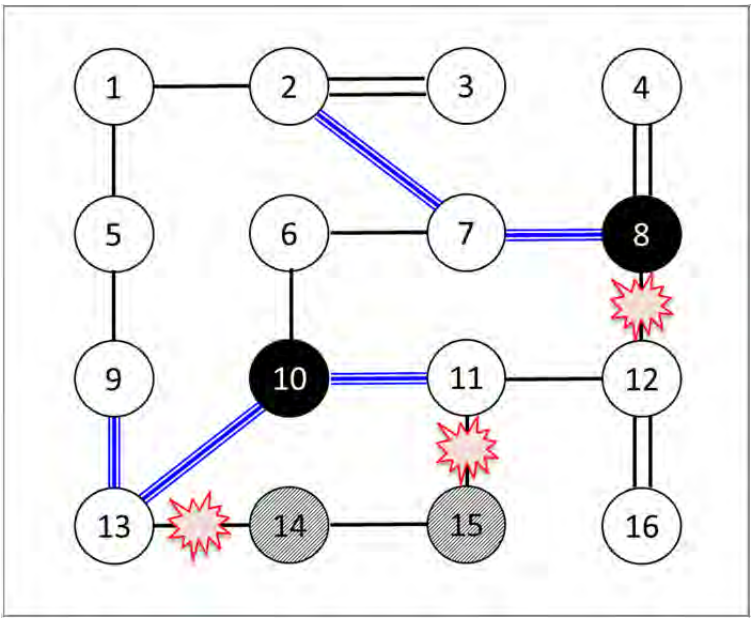

Figure 6. Worst-case disruption with 3 edges targeted and 5 edges protected.

D. EXPECTATION MODEL

We use the symbol $\tilde{x}$ to represent a random event in place of $\hat{x}$, which denotes a set of edges that have been damaged. We calculate the expected cost due to random disruptions, and will not focus on a single, worst-case event. For example, $\Xi \equiv \left\{ x \middle| \sum x_i \le 2 \right\}$ represents the set of all events with two or fewer links broken. For any element $\tilde{x} \in \Xi$, let $P_{\tilde{x}}$ be the probability that $\tilde{x}$ occurs, i.e., $\{ 0 \le P_{\tilde{x}} \le 1 \quad \forall \tilde{x} \in \Xi$, $\sum_{\tilde{x} \in \Xi} P_{\tilde{x}} = 1 \}$. For simplicity, we assume the probability that an edge will go down is independent between edges, giving it a binomial distribution with $\tilde{x}$ becoming a random variable. We can think of Ξ as a list of the enumerated (all) possible attacks with associated probabilities.

Additional Sets

$\xi \in \Xi$ set of (enumerated) outcomes

Additional data [units]

$\tilde{x}^{\xi}$ $\subseteq \{0,1\}^{|E|}$ set of binary vectors, one element for each edge, where $\tilde{x}_{ij}^{\xi} = 1$ if edge $[i, j] \in E$ is damaged, and =0 if the edge remains intact, in outcome $\xi \in \Xi$.

Formulation

$Q(\hat{w}) \equiv E_{\xi}\left(F(\hat{w}, \xi)\right)$, where

$$F(\hat{w}, \xi) \equiv \min_{Y,S} \sum_{d \in D} \sum_{[i,j] \in E} \left[(c_{ij} + q_{ij}^{d} \tilde{x}_{ij}^{\xi}) Y_{ij}^{d} + (c_{ji} + q_{ji}^{d} \tilde{x}_{ij}^{\xi}) Y_{ji}^{d} \right] + \sum_{n \in N} v_n S_n \qquad (\text{E0})$$

s.t. (D3)(D4)

$$\sum_{d \in D} \sum_{j:(n,j) \in A} Y_{nj}^{d} - \sum_{d \in D} \sum_{i:(i,n) \in A} Y_{in}^{d} - S_n \le b_n \quad \forall n \in N \qquad \left[\pi_n^{\xi}\right] \quad (\text{E1})$$

$$Y_{ij}^{d} + Y_{ji}^{d} \le u_{ij} \hat{w}_{ij}^{d} \qquad \forall [i,j] \in A, d \in D \qquad \left[\mu_{ij}^{d\xi}\right] \quad (\text{E2})$$

Whereas the AD model yields both the worst-case attack and associated best-response in the objective value, the result of the ED model is solely the objective value.

Defending against the average case disruption means we must take into account the arcs most likely to fail and either strengthen or build around those arcs in each $\tilde{x}$.

E. DEFENDING AGAINST THE EXPECTED CASE

Just like in the DAD model, we are defending edges in order to minimize the system cost in the presence of disruption by carefully choosing which edges to protect.

The possible defenses will be the same as in the DAD model, just like in Figure 7. However, the result will be the objective value comprised of the products from this design and all possible failures in Ξ creating the *defender-expectation-defender (DED)* case.

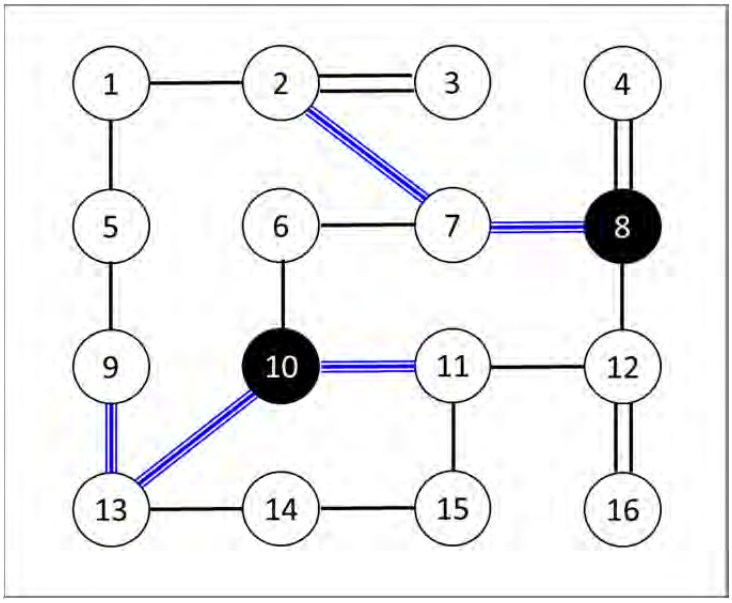

Figure 7. Example Defense in which edges [2,7], [7,8], [9,13], [10,11], and [10,13] are defended.

Formulation

In order to determine the best defense for the expected value problem, we use Benders decomposition (Benders, 1962), an iterative algorithm in which we use the index k to denote an iteration, and K to denote the current iteration of the algorithm. The subproblem for any fixed defense $\hat{w}$ and any specific outcome $\xi \in \Xi$ is just $F(\hat{w}, \xi)$, and can be solved as a linear program with associated duals $\left(\pi_n^{\xi}, \mu_{ij}^{\xi d}\right)$ as given in (E1) and (E2). Each of these dual solutions provides a linear lower bound on the objective function value of $F(\bullet, \xi)$ as a function of the defense:

$$F(\bullet, \xi) \geq \sum_{n \in N} b_n \pi_n^{\xi} + \sum_{[i,j] \in E} \sum_{d \in D} u_{ij} \mu_{ij}^{d\xi} W_{ij}^{d} \ .$$

The weighted sum of these linear functions is thus a lower bound on the *expected* cost as a function of defense, so we then define:

$$\pi_n^k \equiv \sum_{\xi \in \Xi} p_{\xi} \pi_n^{\xi}$$

and

$$\mu_{ij}^{dk} \equiv \sum_{\xi \in \Xi} p_{\xi} \mu_{ij}^{\xi d}$$

to be the weighted combinations of the optimal dual solutions to each of the subproblems

$F\left(\hat{w}^k, \xi\right)$ solved in iteration *k*, yielding a single cut for the set of constraints (SM1) in the master problem. We therefore have the following stochastic master problem:

$$\min_{Z,W} \quad Z_{\text{stoch}} \qquad (\text{SM0})$$

$$\text{s.t.} \quad Z_{\text{stoch}} \geq \sum_{n \in N} b_n \pi_n^k + \sum_{[i,j] \in E} \sum_{d \in D} u_{ij} \mu_{ij}^{dk} W_{ij}^d \quad k = 1, \ldots, K \qquad (\text{SM1})$$

$$\sum_{d \in D} W_{ij}^d = 1 \quad \forall [i, j] \in E \qquad (\text{SM2})$$

$$\sum_{[i,j] \in E} W_{ij}^1 \leq \textit{defense\_budget} \qquad (\text{SM3})$$

The objective (SM0) represents the expected cost of operating in the face of random events. We minimize this objective through each iteration *k* as shown in (SM1). Each edge [*i*,*j*] has two possibilities, denoted with *d*, and only one can be used at a time shown in constraint (SM2). The constraint (SM3) ensures that the defenses used are limited to *defense_budget*. We describe the procedure for solving this problem in the context of the combined model, which now follows.

F. COMBINED MODEL

If we define wt_{stoch} to be the weight given to the stochastic (i.e., expected-value) objective, where $0 \leq wt_{\text{stoch}} \leq 1$, and $wt_{\text{det}} \equiv 1 - wt_{\text{stoch}}$ to be the remaining weight placed on the deterministic (i.e., worst-case) objective, then we can formulate the following combined master problem:

$$\min_{Z,W} \quad wt_{\text{stoch}} Z_{\text{stoch}} + wt_{\text{det}} Z_{\text{det}} \qquad \text{(CM0)}$$

$$s.t. \quad (\text{SM1}), (\text{SM2}), (\text{SM3})$$

$$Z_{\text{det}} \geq \sum_{d \in D} \sum_{[i,j] \in E} \left[(c_{ij} + q_{ij}^d \hat{x}_{ij}^k) Y_{ij}^d + (c_{ji} + q_{ji}^d \hat{x}_{ij}^k) Y_{ji}^d \right]$$

$$+ \sum_{n \in N} v_n S_n^k \qquad k = 1, \ldots, K \qquad (\text{CM2})$$

$$\sum_{d \in D} \sum_{j:[n,j] \in A} Y_{nj}^{dk} - \sum_{d \in D} \sum_{i:[i,n] \in A} Y_{in}^{dk} - S_n^k \leq b_n \quad \forall n \in N, k = 1, \ldots, K \qquad (\text{CM3})$$

$$Y_{ij}^{dk} + Y_{ji}^{dk} \leq u_{ij} W_{ij}^d \qquad \forall [i,j] \in E, d \in D, k = 1, \ldots, K \qquad (\text{CM4})$$

$$Y_{ij}^{dk} \geq 0 \qquad \forall (i,j) \in A, d \in D, k = 1, \ldots, K \qquad (\text{CM5})$$

$$S_n^k \geq 0 \qquad \forall n \in N, k = 1, \ldots, K \qquad (\text{CM6})$$

Here the defense decisions, *W*, are made before any realization of an attack or a random failure, and the evaluation of the defense chosen at iteration *k* is a convex combination of the evaluation of each of the two separate subproblems.

At each iteration *K* of Benders decomposition, our algorithm solves each of the two subproblems (stochastic and deterministic) to optimality for a fixed defense, $\hat{w}^K$. The constraints from the stochastic master problem transfer directly to this combined master problem. The deterministic subproblem yields a single worst-case attack, $\hat{x}^K$, and this provides a lower bound on the cost of the optimal response for that particular attack, although only in terms of the flow variables in response to that attack, and this contributes a new cut to the set of constraints (CM2) in the master problem. We therefore must also create a block of nonnegative variables, Y^K, to model that response, and two additional blocks of constraints to ensure they (a) conform to balance of flow (CM3), and thus act like admissible flows in our network, and (b) correspond appropriately to any defense, W (CM4). The algorithm then solves the master problem for iteration *K*, and generates a new defense, $\hat{w}^{k+1}$. After the subproblems have been solved, we have an accurate evaluation of the combined cost of the current defense plan, $\hat{w}^K$. If this cost is lower than any seen so far, we record $\hat{w}^K$ as a new incumbent solution and take its cost

as the new upper bound. Each solve of the master problem provides a lower bound on the optimal solution, and these values are monotonically nondecreasing. If, at any point, the relative difference between the best upper bound derived from the subproblems and the best lower bound from the master problem is less than a predetermined optimality gap, ε, the algorithm terminates and reports the current incumbent as an ε-optimal solution. This combined model uses the duals from the stochastic model and combines them with the worst case model to produce cuts.

V. RESULTS AND ANALYSIS

In this chapter, we explore the differences between the "worst-case" and "most likely" disruptions for the notional fuel distribution network in Figure 1. We use the combined model to perform our analysis, where in equation (CM0) we have convex weights $wt_{det} = \alpha$ and $wt_{stoch} = 1 - \alpha$.

A. ANALYSIS OF FIVE SCENARIOS

We ran the combined model for five scenarios, each described in the subsections below.

1. Two Attacks and One Defense

Figure 8 displays the results of the combined model for our notional fuel system in the case of two attacks and one defense. For different values of α, we solve the combined model to within an optimality tolerance of 0.001. On a computer with a dual-core 1.2 GHz processor, the solution time for all scenarios involving two attacks was approximately 2 hours for all eleven cases (α=0.0 through α=1.0). For each value of α, Figure 8 shows the value for Z_{stoch} and $Z_{\det}$ along with their combination weighted by α .

When defending solely against the worst-case attack (α=1), the best defense is [2,7] and the worst two-link attack is [7,8] and [8,12]. When defending solely against random events (α=0), the best defense is also [2,7]. Note that the value $Z_{\det}$ is significantly higher than Z_{stoch}, meaning that the worst-case disruption is significantly worse than the average case. Because the worst-case solution does not vary with α, the values for Z_{stoch} and $Z_{\det}$ are insensitive to changes in α. As a result, the combined objective value is not meaningful on its own, rather it represents simply the weighted combination of these two constant values.

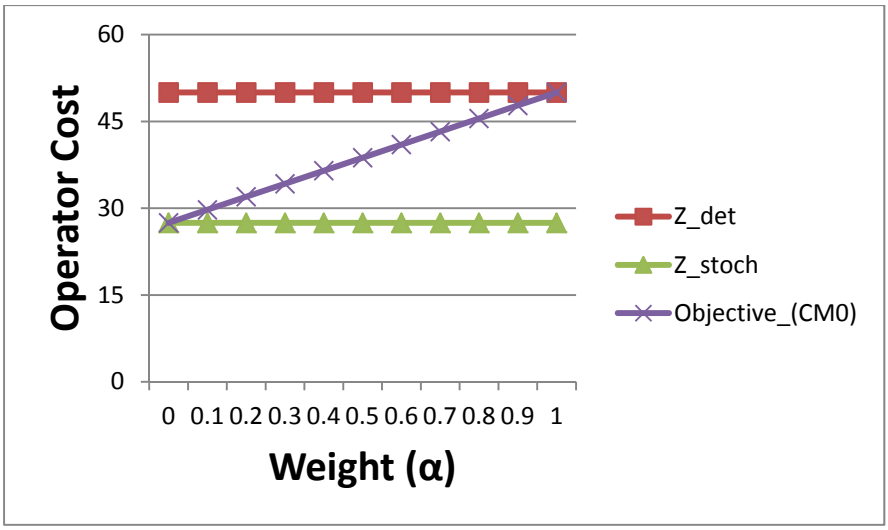

Figure 8. The objective values for two attacks and one defense versus the weight for given weight α.

2. Two Attacks and Two Defenses

Just as in the previous case, the worst-case solution does not vary with α, the values for Z_{stoch} and $Z_{\det}$ are insensitive to changes in α. When defending solely against the worst-case attack (α=1), the best defense is [2,7] and [7,8]; while, the worst two-link attack is [1,2] and [9,13]. When defending solely against random events (α=0), the best defense is also [2,7] and [7,8].

Table 1 summarizes the results from the scenarios involving no more than two broken links.

Table 1. Optimal links to defend for different defensive budgets for cases when there are no more than two broken links. For this small network, we observe that the single best link to protect is [2,7], and when defending two links the best two are [2,7] and [7,8].

Def	Stoch (α =0)	Combined (α =.5)	Det (α=1)
1	[2,7]	[2,7]	[2,7]
2	[2,7][7,8]	[2,7][7,8]	[2,7][7,8]

3. Three Attacks and One Defense

The results of the combined model for our notional fuel system in the case of three attacks and one defense has a shape similar to the two-attack case. For this scenario, we solve the combined model for different values of α to within an optimality tolerance of 0.1. On a computer with a dual-core 1.2 GHz processor, the solution time for was approximately 6 hours for all eleven cases (α=0.0 through α=1.0). Table 2 displays the results of this computation.

Table 2. Combined model results for three attacks and one defense

α	edge to defend	worst attack	objective_(CM0)	obj_Zdet	obj_Zstoch
0.0 (Stoch)	[2,7]	[9,13][6,10][7,8]	29.47955282	80	29.47955282
0.1	[2,7]	[9,13][6,10][7,8]	34.53159754	80	29.47955282
0.2	[2,7]	[9,13][6,10][7,8]	39.58364226	80	29.47955282
0.3	[2,7]	[9,13][6,10][7,8]	44.63568697	80	29.47955282
0.4	[2,7]	[9,13][6,10][7,8]	49.68773169	80	29.47955282
0.5	[2,7]	[9,13][6,10][7,8]	54.73977641	80	29.47955282
0.6	[2,7]	[9,13][6,10][7,8]	59.79182113	80	29.47955282
0.7	[2,7]	[9,13][6,10][7,8]	64.84386585	80	29.47955282
0.8	[2,7]	[9,13][6,10][7,8]	69.89591056	80	29.47955282
0.9	[13,10]	[9,13][6,10][7,8]	75.02040504	80	30.2040504
1.0 (Det)	[13,10]	[9,13][6,10][7,8]	80	80	30.2040504

When defending solely against the worst-case attack (α=1), the best defense is [13,10] and the worst three-link attack is [7,8], [9,13] and [6,10] as shown in Table 2. While defending solely against random events (α=0), the best defense is [2,7]. There is a difference in the solution for the worst-case and the solution for random events; however, the solution doesn't change until $\alpha \geq 0.9$ toward the worst-case. The difference in cost to the operator between protecting edge [2,7] and [13,10] is one. Similar to the two attack scenarios, the value Z_{det} is significantly higher than Z_{stoch}. There is more value in protecting [2,7] for the stochastic case than [13,10], but the values for Z_{stoch} are similar for the two cases. Overall, the values for Z_{stoch} and Z_{det} are insensitive to changes in α, but

are far from each other. Again, the combined objective value represents simply the weighted combination of these two constant values.

4. Three Attacks and Two Defenses

We similarly solve for the results of the combined model for our notional fuel system in the case of three attacks and two defenses. On a computer with a dual-core 1.2 GHz processor, the solution time for was approximately 10 hours for all eleven cases (α=0.0 through α=1.0). Table 3 displays the results of this computation.

Table 3. Combined model results for three attacks and two defenses

α	edge to defend	worst attack	objective_(CM0)	obj_Zdet	obj_Zstoch
0.0 (Stoch)	[2,7][7,8]	[1,2][13,10][11,15]	28.03713818	72	28.03713818
0.1	[2,7][7,8]	[1,2][13,10][11,15]	32.43342436	72	28.03713818
0.2	[13,10][7,8]	[13,14][8,12][10,11]	36.47357923	67	28.84197404
0.3	[13,10][7,8]	[13,14][8,12][10,11]	40.28938182	67	28.84197404
0.4	[13,10][7,8]	[13,14][8,12][10,11]	44.10518442	67	28.84197404
0.5	[13,10][7,8]	[13,14][8,12][10,11]	47.92098702	67	28.84197404
0.6	[13,10][7,8]	[13,14][8,12][10,11]	51.73678961	67	28.84197404
0.7	[13,10][7,8]	[13,14][8,12][10,11]	55.55259221	67	28.84197404
0.8	[13,10][7,8]	[13,14][8,12][10,11]	59.36839481	67	28.84197404
0.9	[13,10][7,8]	[13,14][8,12][10,11]	63.1841974	67	28.84197404
1.0 (Det)	[13,10][7,8]	[13,14][8,12][10,11]	67	67	28.84197404

The difference between defending solely against random events (α=0) and the worst-case attack (α=1) is the defending of edge [2,7] vice [13,10]. Although [13,10] and [2,7] yield the two highest operating costs for attacking a single edge, it is more important to protect only one of them along with [7,8] due the construction of the network as shown in Figure 1. Unlike the previous defense for three attacks, the solution turns to the worst-case when $\alpha \geq 0.2$, meaning, when coupled with the defense of [7,8] it is more beneficial for both models to protect [13,10] vice [2,7].

5. Three Attacks and Three Defenses

We similarly solve for the results of the combined model for our notional fuel system in the case of three attacks and three defenses. On a computer with a dual-core 1.2

GHz processor, the solution time for was approximately 13 hours for all eleven cases (α=0.0 through α=1.0). Table 4 displays the results of this computation.

Table 4. Combined model results for three attacks and three defenses

α	edge to defend	worst attack	objective_(CM0)	obj_Zdet	obj_Zstoch
0.0 (Stoch)	[2,7][7,8][9,13]	[1,2][13,10][11,15]	27.45686501	72	27.45686501
0.1	[2,7][7,8][13,10]	[8,12][13,14][10,11]	31.26976353	67	27.29973726
0.2	[2,7][7,8][13,10]	[8,12][13,14][10,11]	35.23978981	67	27.29973726
0.3	[13,10][7,8][8,12]	[2,7][9,13][10,11]	39.12518556	64	28.4645508
0.4	[13,10][7,8][8,12]	[2,7][9,13][10,11]	42.67873048	64	28.4645508
0.5	[13,10][7,8][8,12]	[2,7][9,13][10,11]	46.2322754	64	28.4645508
0.6	[13,10][7,8][8,12]	[2,7][9,13][10,11]	49.78582032	64	28.4645508
0.7	[13,10][7,8][8,12]	[2,7][9,13][10,11]	53.33936524	64	28.4645508
0.8	[13,10][7,8][8,12]	[2,7][9,13][10,11]	56.89291016	64	28.4645508
0.9	[13,10][7,8][8,12]	[2,7][9,13][10,11]	60.44645508	64	28.4645508
1.0 (Det)	[13,10][7,8][10,11]	[2,7][9,13][8,12]	64	64	28.68990798

The three attack and three defense solution for the combined model isn't as intuitive as the previous two scenarios. When $0.3 \le \alpha \le 0.9$, the optimal solution defends edges [8,12], [13,10], and [7,8]. Showing that moderate protection against both random and worst-case disruptions is separate from the solutions for either event as shown in Table 2.

Figure 9 shows that as the values for the separate models change, the objective function still maintains the linear characteristic as expected; however the operator costs for Z_{det} goes down for each change in defense while Z_{stoch} goes up slightly for the increase in α. The worst case and expectation models both chose the same edges to defend for the two attack scenario as shown in the Table 1. This result makes it easy for a decision maker to choose which items to defend for the best possible protection. The problem for decision makers is shown in the case of three attacks.

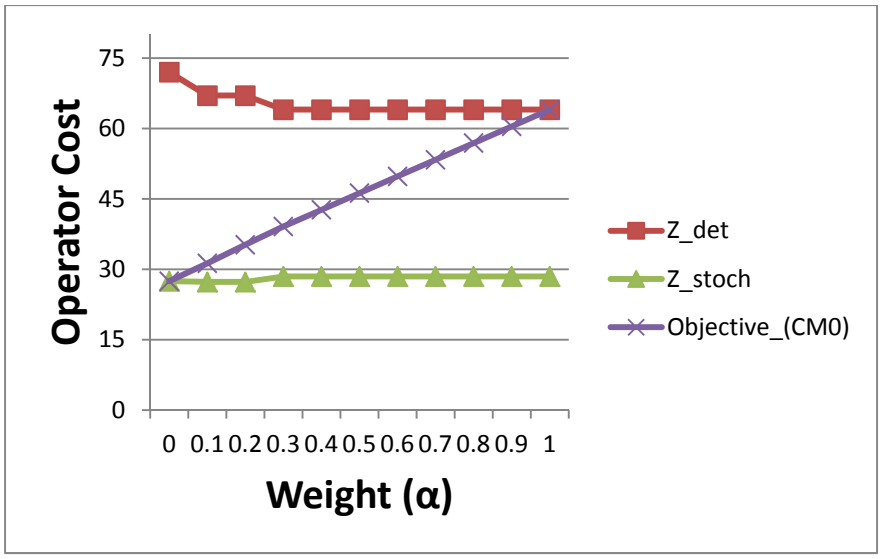

Figure 9. The objective values for three attacks and three defenses versus the weight for given weight α.

B. DISCUSSION

In the case of random disruptions, with a limited number of defenses, we might not be able to protect our system against the (potentially enormous) set of disruptions near the "center" of the distribution. However, if there are only a few disruptions that cause significant damage (i.e., well above the mean) then we can apply our limited resources to reduce their impact and therefore have a noticeable effect on the expected cost over the entire distribution of random disruptions.

Figure 10 displays the overall value of having one, two, or three defenses, when protecting against three attacks. There are diminishing returns for the third defense used. To save money the decision maker should use a two defense approach vice three in this case.

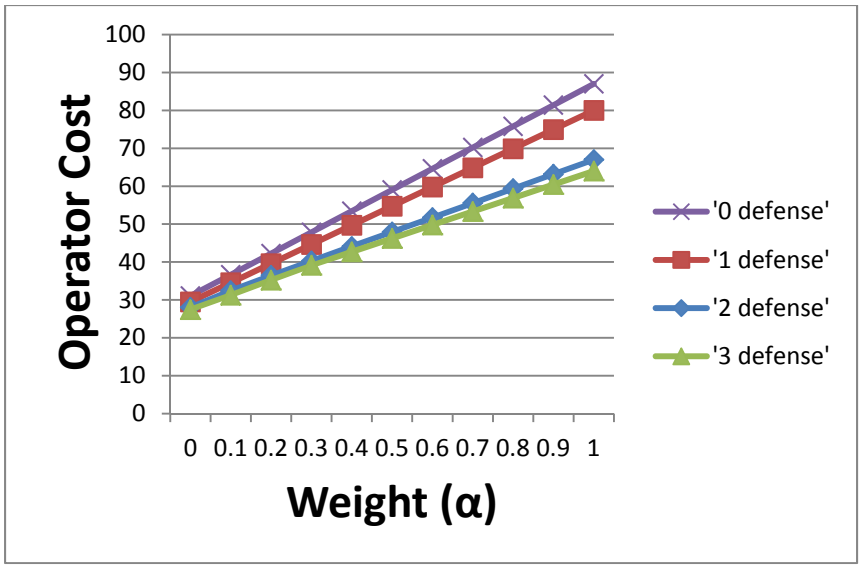

Figure 10. Multiple defense options against the three attacks scenario: Objective value for given defense with weight α in equation (CM0), where wt_{det} = α and wt_{stoch} = 1 – α.

THIS PAGE INTENTIONALLY LEFT BLANK

VI. CONCLUSIONS

In this thesis, we have presented a model for defensive investment that combines the threat of a worst-case disruption with the risk of a random disruption. Our combination model has the potential to help decision makers come up with a good defense that fits their particular network. From our analysis we discover that it is possible to choose the best defense for that given network by mapping the frontier between the worst-case and random disruptions.

Our investigation of a small network system provides proof-of-concept that the technique we propose can be effective, but there is more work to be done. We discovered that the stochastic model runs longer than the combined model. Interestingly, this suggests that knowing how to defend against the worst-case potentially provides insight into how to defend against random events. However, even for the small example here, the program takes a long period of time to solve the combined model for three attacks and two or more defenses. Therefore, additional research to reduce the required computational effort is needed before the technique can be used at large scale.

THIS PAGE INTENTIONALLY LEFT BLANK

APPENDIX

Our computational models and solution were obtained with the Python programming language and the Cooper-Pyomo optimization module using CPLEX as the solver. The program is solved using five modules. The `combo_main.py` module contains the outer loop for multiple runs of the model and is the driver that branches into the other modules. We have a separate module, `phat_code.py`, to create the distribution for the probability of damaging one or more arcs. The network is built using the networkx module, in `netx_graph.py`, by building the list of edges and nodes. The file `combo_model.py` creates the pyomo models for the master and sub problems used to create cuts and implement benders. The file `pyomo_model.py` creates and solves the minimum cost flow model. The code for the modules are listed below:

```
#Begin combo_main.py
import netx_graph
import pyo_model
import combo_model
import phat_code
import matplotlib.pyplot as plt
#open file
file= open('ComboResults.txt','w')
file.write('Weight (Worst), Arcs Defended, Objective value, Worst attack,Worst value,Random value\n')
test = [0,0.1,0.2,0.3,0.4,0.5,0.6,0.7,0.8,0.9,1]
for weight in test:
    if __name__ == '__main__':
        #establish nodes, edges, start_nodes, and double_edges
        #This will eventually occur in a subroutine that parses input files
        nodes = ['1','2','3','4','5','6','7','9','11','12','13','14','15','16']
        edges=
[('1','2'),('2','7'),('1','5'),('5','9'),('9','13'),('13','10'),('13','14'),('14','15'),('6','10'),('6','7'),('7','8'),('8','12'),('10','11'),('11','12'),('11','15')]
        start_nodes = ('10','8')
        double_edges =[('2','3'),('8','4'),('12','16')]
        #build graph
        Grph, vuln, double_nodes = netx_graph.newmodel(nodes, start_nodes, edges, double_edges)
```

```
#establish single-edge probabilities and initialize xhat
phat, xhat = phat_code.phat_edges(Grph, vuln)
defenses = ['0', '1']
w_hat = {}
for (i,j) in vuln:
    w_hat[i,j,'0'] = 1
    w_hat[i,j,'1'] = 0

#Create dictionaries for node supplies and vulnerable edge capacities
supply = {}
for item in Grph.nodes():
    if item in start_nodes:
        supply[item] = 10
    elif item in double_nodes:
        supply[item] = 0
    else:
        supply[item] = -1
capacity = {}
for (i,j) in vuln:
    capacity[i,j] = 10
#generate probability map for all events with two or fewer failures
P = phat_code.create_Prob(Grph,vuln,phat,3)
print 'Number of events: ' + str(len(P))
max_iterations = 50
nodepi = {}  #dictionary of node duals, by iteration
edgemu = {}  #dictionary of arc capacity duals, by iteration
xhats = {}   #dictionary of worst-case attacks, by iteration

LB=0               # can set this based on first MP solve...
UB=10000           # should come up with a better way of handling this
master_tol = 0.10  # relative gap tolerance for terminating benders
det_wt = weight
stoch_wt = 1-weight
for iteration in range(max_iterations):
    expected_cost = 0
    cumul_p = 0
    for (scen_num,x_tilde) in enumerate(P.keys()):
        xhat = phat_code.generate_xhat_from_x_tilde(vuln,x_tilde,xhat)
```

```
                model,                          instance                        =
pyo_model.build_model(Grph,xhat,w_hat,vuln,defenses,supply,capacity)
                results = pyo_model.solve_model(instance)
                instance.load(results)
                objective_value = instance.FlowCost()
                if scen_num == 0:
                    worst_scen = x_tilde
                    worst_obj = objective_value
                    for node in Grph.nodes():
                        dualval                                                 =
instance.dual.getValue(instance.FlowBalance[node])
                        if abs(dualval) > 1e-08:
                            nodepi[node,iteration] = P[x_tilde]*dualval
                        else:
                            nodepi[node,iteration] = 0.0
                    for (i,j) in vuln:
                        for defense in defenses:
                            dualval=
instance.dual.getValue(instance.EdgeCapacity[i,j,defense])
                            if abs(dualval) > 1e-08:
                                edgemu[i,j,defense,iteration]=
P[x_tilde]*dualval
                            else:
                                edgemu[i,j,defense,iteration] = 0.0
                else:
                    if worst_obj < objective_value:
                        worst_obj = objective_value
                        worst_scen = x_tilde
                    for node in Grph.nodes():
                        dualval=
instance.dual.getValue(instance.FlowBalance[node])
                        if abs(dualval) > 1e-08:
                            nodepi[node,iteration] += P[x_tilde]*dualval
                    for (i,j) in vuln:
                        for defense in defenses:
                            dualval=
instance.dual.getValue(instance.EdgeCapacity[i,j,defense])
                        if abs(dualval) > 1e-08:
                            edgemu[i,j,defense,iteration] += P[x_tilde]*dualval


                expected_cost += P[x_tilde] * objective_value
```

```
                cumul_p += P[x_tilde]
                #provide intermittent updates
                #if scen_num % 25 == 0:
                #    print str(scen_num) + ': ' + str(objective_value) + '
approx. expected cost: ' + str(expected_cost/cumul_p)
                #if len(x_tilde)==0:
                #  print '***' + str(scen_num) + ': ' + str(objective_value) +
'  approx. expected cost: ' + str(expected_cost/cumul_p)
            print 'Subproblem Iteration:' + str(iteration)
            print '   Expected Cost   = ' + str(expected_cost)
            print '   Worst-Case Cost = ' + str(worst_obj)
            for (i,j) in worst_scen:
                print '      Attack: (' + str(i) + ', ' + str(j) + ')'
            print '     Weighted Cost     = ' + str(stoch_wt*expected_cost +
det_wt*worst_obj)
            #print edgemu
            #print nodepi
            if iteration==0:
                incumbent = w_hat
                UB = stoch_wt*expected_cost + det_wt*worst_obj
                inc_iteration = 0
                print '***new incumbent: (UB) ' + str(UB)
                x_star=worst_scen
                x_star_obj= worst_obj
                x_tilde_obj=expected_cost
            else:
                if UB > stoch_wt*expected_cost + det_wt*worst_obj:
                    UB = stoch_wt*expected_cost + det_wt*worst_obj
                    incumbent=w_hat
                    inc_iteration = iteration
                    print '***new incumbent: (UB) ' + str(UB)
                    x_star=worst_scen
                    x_star_obj= worst_obj
                    x_tilde_obj=expected_cost
            if UB-LB<master_tol*LB:
                break
            for (i,j) in vuln:
                if (i,j) in worst_scen:
                    xhats[i,j,iteration] = 1
                else:
```

```
                    xhats[i,j,iteration] = 0


            model, instance = combo_model.BuildComboMaster(Grph, vuln,
defenses, 3, iteration+1, det_wt, stoch_wt, xhats, nodepi, edgemu, supply,
capacity)
            results = combo_model.SolveModel(instance)
            instance.load(results)
            print 'Master Iteration: ' + str(iteration)
            w_hat = {}
            for (i,j) in vuln:
                for d in defenses:
                    if instance.W[i,j,d].value > 1e-08:
                        w_hat[i,j,d] = 1
                        if d=='1':
                            print 'Defend (' + str(i) + ',' + str(j) + ')  W='
+ str(w_hat[i,j,d])
                    else:
                        w_hat[i,j,d] = 0
            if LB < instance.MasterObjective():
                LB = instance.MasterObjective()
            print 'Master objective: (LB) ' + str(instance.MasterObjective())
            if UB-LB<master_tol*LB:
                break
        print
        if UB-LB<master_tol*LB:
            print 'Optimality Gap Reached at iteration=' +str(iteration)
        else:
            print 'Iteration limit reached at iteration=' +str(iteration)
        print 'Incumbent Defense found at iteration=' +str(inc_iteration)
        for (i,j) in vuln:
            if incumbent[i,j,'1'] > 1e-08:
                print '  Defend (' + str(i) + ',' + str(j) + ')'
                file.write('('+str(i) + ',' + str(j) + ')' )
        print 'Expected Cost:  ' + str(UB)
        print 'Lower Bound:    ' + str(LB)
        if LB>0:
            print 'Optimality gap: ' + str((UB-LB)/LB)
file.write(str(weight)+','+str(UB)+','+str(x_star)+','+str(x_star_obj)+','+str(
x_tilde_obj)+'\n')
file.close()
#End combo_main.py
```

```
#Begin netx_graph.py
#Build the graph using networkx
import networkx as nx

def twolines(a,b,grph, vulnerable_edges, double_nodes):
    #create the equivalent of two parallel edges between nodes a and b by
adding
    #four artificial nodes and twelve directed arcs
    suffix_1 = "_" + a + "_" + b
    suffix_2 = "_" + b + "_" + a
    grph.add_node(a + suffix_1, node_size = 99,node_color='y')
    grph.add_node(b + suffix_1, color='y')
    grph.add_node(a + suffix_2, color='y')
    grph.add_node(b + suffix_2, color='y')
    grph.add_edge(a, a + suffix_1, cost =0)
    grph.add_edge(a, a + suffix_2, cost =0)
    grph.add_edge(a + suffix_1, a, cost =0)
    grph.add_edge(a + suffix_2, a, cost =0)
    grph.add_edge(b, b + suffix_1, cost=0)
    grph.add_edge(b, b + suffix_2, cost=0)
    grph.add_edge(b + suffix_1, b, cost=0)
    grph.add_edge(b + suffix_2, b, cost=0)
    grph.add_edge(a + suffix_1, b + suffix_1, cost = 1)
    grph.add_edge(a + suffix_2, b + suffix_2, cost = 1)
    grph.add_edge(b + suffix_1, a+suffix_1, cost = 1)
    grph.add_edge(b + suffix_2, a+suffix_2, cost = 1)
    vulnerable_edges.append( (a+suffix_1,b+suffix_1) )
    vulnerable_edges.append( (a+suffix_2,b+suffix_2) )
    double_nodes.append(a+suffix_1)
    double_nodes.append(a+suffix_2)
    double_nodes.append(b+suffix_1)
    double_nodes.append(b+suffix_2)
    return

def doubs(a, b, grph, vulnerable_edges):
    #create two directed arcs to represent an undirected edge between nodes a
and b
    grph.add_edge(a, b, cost=1)
    grph.add_edge(b, a, cost=1)
    vulnerable_edges.append( (a,b) )
    return

#Create the model
def newmodel(nodes, start_nodes, edges, double_edges):
    vulnerable_edges = []
    double_nodes = []
    g = nx.DiGraph()
    #node list
    for i in nodes:
        if i in start_nodes:
            g.add_node(i, node_color ='b')
        else:
            g.add_node(i, node_color='r')

    #edge list
    for i,j in edges:
        doubs(i,j,g,vulnerable_edges)
    for i,j in double_edges:
        twolines(i,j,g,vulnerable_edges, double_nodes)
    return g, vulnerable_edges, double_nodes
#End netx_graph.py
```

```
#Begin phat_code.py
#Import random functions
import random
from itertools import combinations
import math

#Take a list of vulnerable edges and assign probabilities to them
def phat_edges(graph,vulnerable_edges):
    random.seed(0)
    phat = {}
    xhat = {}
    #Initialize probablity of arcs with random and initialize xhat =0
    #for tups in graph.edges():
    for tups in vulnerable_edges:
        temp = random.random()
        phat[tups]= temp*.3 + .01
        xhat [tups]= 0
    return phat, xhat

def create_Prob(graph,vulnerable_edges,phat,max_losses=None):
    """Build dictionary of events and their associated probabilities.
    Assumes elements (vulnerable_edges) fail independently with
    probabilities specified by the dictionary phat. If
    max_losses is specified, computes conditional probabilities
    for events whose cardinality does not exceed it."""
    Prob={}
    if max_losses == None or max_losses > len(vulnerable_edges):
        max_losses = len(vulnerable_edges)
    for i in range(max_losses+1):
        for x_tilde in combinations(vulnerable_edges,i):
            p=1
            for e in vulnerable_edges:
                if e in x_tilde:
                    p *= phat[e]
                else:
                    p *= 1-phat[e]
            Prob[x_tilde]=p
    #normalize probabilities
    totprob = math.fsum(Prob.values())
    for x_tilde in Prob:
        Prob[x_tilde] /= totprob
    return Prob


def generate_xhat(vulnerable_edges,phat, xhat):
    for edges in vulnerable_edges:
        temp = random.random()
        if temp <= phat[edges]:
            xhat[edges]=1
        else:
            xhat[edges]=0
    return xhat

def generate_xhat_from_x_tilde(vulnerable_edges,x_tilde,xhat):
    for edge in vulnerable_edges:
        if edge in x_tilde:
            xhat[edge]=1
        else:
            xhat[edge]=0
    return xhat
#End phat_code.py
```

```
#Begin combo_model.py
from __future__ import division
import sys

#Import coopr and Solver
from coopr.pyomo import *
from coopr.opt import SolverFactory

#Master cuts
def MasterStochCut(model,iter_num):
    nodeterm = sum(model.b[i]*model.pi[i,iter_num] for i in model.Nodes)
    arcterm = sum(model.u[i,j]*model.mu[i,j,d,iter_num]*model.W[i,j,d] for
(i,j,d) in model.Vuln*model.Defenses)
    return model.Z_Stoch >= nodeterm + arcterm

def MasterDetCut(model,iter_num):
    edge_sum = (
sum((model.Y[i,j,'0',iter_num]*(model.c[i,j]+(model.xhat[i,j,iter_num]*model.q[
i,j])))for (i,j) in model.Vuln)

+sum((model.Y[j,i,'0',iter_num]*(model.c[j,i]+(model.xhat[i,j,iter_num]*model.q
[j,i])))for (i,j) in model.Vuln)
                +sum((model.Y[i,j,'1',iter_num]*(model.c[i,j])) for (i,j) in
model.Arcs)
                 )
    node_sum = sum(model.v[j]*model.S[j, iter_num] for j in model.Nodes)
    return  model.Z_Det >= edge_sum + node_sum

def MasterDetFlowBalance(model,node,iter_num):
    amountIn = sum (model.Y[i,j,defense,iter_num] for (i,j,defense) in
model.Arcs*model.Defenses if j==node)
    amountOut = sum (model.Y[i,j,defense,iter_num] for (i,j,defense) in
model.Arcs*model.Defenses if i==node)
    supply = model.b[node]
    return amountOut - amountIn - model.S[node,iter_num] <= supply

def MasterDetEdgeCapacity(model,i,j,defense,iter_num):
    return model.Y[i,j,defense,iter_num] + model.Y[j,i,defense,iter_num] <=
model.u[i,j]*model.W[i,j,defense]

#Choose one defense for each vulnerable edge
def MasterOneDefense(model,i,j):
    return sum(model.W[i,j,d] for d in model.Defenses) == 1

#Limit number of defenses
def MasterDefenseLimit(model):
    return sum(model.W[i,j,'1'] for (i,j) in model.Vuln) <= model.max_defenses

#Master Objective
def MasterObjective(model):
    return model.det_wt*model.Z_Det + model.stoch_wt*model.Z_Stoch

#Ensure name of networkx graph is Grph to run effectively
# need networkx graph named Grph, dictionary for xhats
def BuildComboMaster(Grph, vulnerable_edges, defenses, max_defenses,
iterations, det_wt, stoch_wt, xhats, nodepi, edgemu, supply, capacity):
    #Define model type
    model = AbstractModel()

    #Put model into Pyomo
    nodes = Grph.nodes()
    model.Nodes = Set(initialize=nodes)
```

```
    model.Vuln = Set(within=model.Nodes*model.Nodes,
initialize=vulnerable_edges)
    arcs = Grph.edges()
    model.Arcs = Set(within=model.Nodes*model.Nodes,initialize=arcs)

    iters = range(iterations)
    model.Iters = Set(initialize=iters)

    model.Defenses = Set(initialize=defenses)

    model.max_defenses = Param(initialize=max_defenses)
    #Node dual parameters
    model.pi = Param(model.Nodes*model.Iters, initialize=nodepi)

    #Arc capacity dual parameters
    model.mu = Param(model.Arcs*model.Defenses*model.Iters, initialize=edgemu)

    model.xhat = Param(model.Vuln*model.Iters, initialize=xhats)

    #arc costs
    values= {}
    for item in Grph.edges():
        values[item] = Grph.edge[item[0]][item[1]]['cost']
    model.c = Param(model.Nodes,model.Nodes, initialize=values)

    #supplies
    model.b = Param(model.Nodes,initialize=supply)

    #capacities
    model.u = Param(model.Vuln, initialize=capacity)

    #Penalty model
    model.q = Param(model.Arcs, initialize=100)

    #Penalty unsupplied nodes
    model.v = Param(model.Nodes, initialize=10)

    model.det_wt = Param(initialize=det_wt)
    model.stoch_wt = Param(initialize=stoch_wt)

    #define objective variables
    model.Z_Stoch = Var(domain=NonNegativeReals)
    model.Z_Det = Var(domain=NonNegativeReals)

    #define design variables
    model.W = Var(model.Vuln*model.Defenses, domain=Binary)

    #define response variables for deterministic cuts
    model.Y = Var(model.Arcs*model.Defenses*model.Iters,
domain=NonNegativeReals)

    #Define fuel shortfall at Node
    model.S = Var(model.Nodes*model.Iters, domain=NonNegativeReals)

     #Master cut constraints
    model.MasterStochCut = Constraint(model.Iters, rule=MasterStochCut)
    model.MasterDetCut = Constraint(model.Iters, rule=MasterDetCut)

    #Det flow constraints for each iter
    model.MasterDetFlowBalance = Constraint(model.Nodes*model.Iters,
rule=MasterDetFlowBalance)
```

```
    model.MasterDetEdgeCapacity =
Constraint(model.Vuln*model.Defenses*model.Iters, rule=MasterDetEdgeCapacity)

    #Master arc defense constraints
    model.MasterOneDefense = Constraint(model.Vuln, rule=MasterOneDefense)

    #Master limit on number of defenses
    model.MasterDefenseLimit = Constraint(rule=MasterDefenseLimit)

    #Objective function
    model.MasterObjective = Objective(rule=MasterObjective, sense=minimize)

    instance=model.create()

    return model, instance

#Create the model instance and Solve returning values for
#min flow path in variable called shortest
#need list of double_nodes to create key and edge labels

def SolveModel(instance):
    opt = SolverFactory("cplex")
    #opt = SolverFactory("glpk")
    #ask for duals and reduced costs from solver
    results = opt.solve(instance, suffixes=['dual','rc'])
    return results

def BuildResults(instance,results):
    instance.load(results)
    #figure arcs defended
    defended = []
    for key in instance.W:
        if instance.W[key].value > 0:
            defended.append(key)

    objective_value = instance.MasterObjective()
    return defended, objective_value
#End combo_model.py
```

```
#Begin pyo_model.py
from __future__ import division
import sys

#Import coopr and Solver
from coopr.pyomo import *
from coopr.opt import SolverFactory

#Network balance of flow constraint
def FlowBalance(model,node):
    amountIn = sum (model.Y[i,j,defense] for (i,j,defense) in
model.Arcs*model.Defenses if j==node)
    amountOut = sum (model.Y[i,j,defense] for (i,j,defense) in
model.Arcs*model.Defenses if i==node)
    supply = model.b[node]
    return amountOut - amountIn - model.S[node] <= supply

#Define objective equation
def FlowCost(model):
    edge_sum = (
sum((model.Y[i,j,'0']*(model.c[i,j]+(model.xhat[i,j]*model.q[i,j])))for (i,j)
in model.Vuln)

+sum((model.Y[j,i,'0']*(model.c[j,i]+(model.xhat[i,j]*model.q[j,i])))for (i,j)
in model.Vuln)
                +sum((model.Y[i,j,'1']*(model.c[i,j])) for (i,j) in model.Arcs)
                 )
    node_sum = sum(model.v[j]*model.S[j] for j in model.Nodes)
    return  edge_sum + node_sum

def EdgeCapacity(model, i, j, defense):
    return model.Y[i,j,defense] + model.Y[j,i,defense] <=
model.u[i,j]*model.w_hat[i,j,defense]

#Ensure name of networkx graph is Grph to run effectively
# need networkx graph named Grph, dictionary for xhats
def build_model(Grph, xhat, w_hat, vulnerable_edges, defenses, supply,
capacity):
    #Define model type
    model = AbstractModel()

    #Put model into Pyomo
    nodes = Grph.nodes()
    model.Nodes = Set(initialize= nodes)

    edges = Grph.edges()
    model.Arcs = Set(within=model.Nodes*model.Nodes,initialize = edges)

    model.Vuln = Set(within=model.Nodes*model.Nodes,initialize =
vulnerable_edges)

    model.Defenses = Set(initialize=defenses)

    #create suffixes for dual information
    model.dual = Suffix(direction=Suffix.IMPORT_EXPORT)

    #Create model for costs
    values= {}
    for item in edges:
        values[item] = Grph.edge[item[0]][item[1]]['cost']
    model.c = Param(model.Nodes,model.Nodes, initialize=values)
```

```
    #Supplies
    model.b = Param(model.Nodes,initialize=supply)

    #Penalty model
    model.q = Param(model.Arcs, initialize=100)

    #Penalty unsupplied nodes
    model.v = Param(model.Nodes, initialize=10)

    #Capacities on vulnerable arcs
    model.u = Param(model.Vuln, initialize=capacity)

    #Define attack variable
    model.xhat = Param(model.Vuln,initialize=xhat)

    #Define attack variable
    model.w_hat = Param(model.Vuln*model.Defenses,initialize=w_hat)

    #Define Variable we want to change
    model.Y = Var(model.Arcs*model.Defenses, domain=NonNegativeReals)

    #Define fuel shortfall at Node
    model.S = Var(model.Nodes, domain=NonNegativeReals)

    #Constraint for balance
    model.FlowBalance = Constraint(model.Nodes, rule=FlowBalance)

    model.EdgeCapacity = Constraint(model.Vuln*model.Defenses,
rule=EdgeCapacity)

    #Objective function
    model.FlowCost = Objective(rule=FlowCost, sense=minimize)
    instance=model.create()

    return model, instance

#Create the model instance and Solve returning values for
#min flow path in variable called shortest
#need list of double_nodes to create key and edge labels

def solve_model(instance):
    opt = SolverFactory("cplex")
    #opt = SolverFactory("glpk")
    #ask for duals and reduced costs from solver
    results = opt.solve(instance, suffixes=['dual','rc'])
    return results

def build_results(instance,results):
    instance.load(results)
    #figure arcs used
    shortest = []
    edge_values = {}
    for key in instance.Y:
        if instance.Y[key].value > 0:
            shortest.append((key[0],key[1]))
            edge_values[(key[0],key[1])] = instance.Y[key].value

    objective_value = instance.FlowCost(instance)
    return shortest, edge_values, objective_value

       #End pyo_model.py
```

LIST OF REFERENCES

Alderson, D., Brown, G., Carlyle, W., Cox, L. (2013) Sometimes there is no "most vital" arc: assessing and improving the operational resilience of systems. *Mil. Op. Res.* 18: 21–37.

Alderson, D., Brown, G., Carlyle, W. (2014a) Assessing and improving operational resilience of critical infrastructures and other systems. (A. Newman and J. Leung, eds. *Tutorials in Operations Research*. INFORMS, Catonsville, MD). Forthcoming.

Alderson, D., Brown, G., Carlyle, W. (2014b) Operational Models of Infrastructure Resilience. *Risk Analysis* Forthcoming.

Bedford, T., Cooke, R. (2001) *Probabilistic Risk Analysis: Foundations and Methods*. **(**Cambridge University Press, New York).

Benders, J. (1962) Partitioning procedures for solving mixed-variables programming problems. *Numerische Mathematik* 4 238–252.

Bier, V.M. (2007) Choosing what to protect. *Risk Analysis* 27(3) 607–620.

Billinton R., Allan, R. (1992) *Reliability Evaluation Of Engineering Systems|Concepts and Techniques,* 2nd ed. (Plenum Press, New York).

Brown, G., Carlyle, M., Salmeron, J., Wood, K. (2005) Analyzing the vulnerability of critical infrastructure to attack and planning defenses, in *Tutorials in Operations Research: Emerging Theory, Methods, and Applications.(* H. Greenberg and J. Smith, eds., Institute for Operations Research and Management Science, Hanover, MD).

Brown, G., Carlyle, M., Salmeron, J., Wood, K. (2006) Defending critical infrastructure. *Interfaces* 36 530–544.

Burton, C. (2013) Analyzing the U.S. military fuel distribution network on Okinawaw. Master's thesis, (Naval Postgraduate School, Monterey, California).

De la Cruz, C. (2011) Defending the maritime transport of cargo for the Hawaiian Islands. Master's thesis, (Naval Postgraduate School, Monterey, California).

Department of Homeland Security (DHS). (2013) *National Infrastructure Protection Plan.*

Department of Justice (DOJ). (2001) *The USA PATRIOT Act: Preserving Life and Liberty*. http://www.justice.gov/archive/ll/what_is_the_patriot_act.pdf

Garrick, B.J., Hall, J., MacDonald, J., OToole, T., Probst, P., Parker, E., Rosenthal, R., Trivelpiece, A., Van Arsdale, L., Zebroski, E. (2004) Confronting the risks of terrorism: *Making the right decisions. Reliability Engineering and System Safety* 86(2) 129–176.

Hausken K., Bier, V., Zhuang, J. (2009) Defending against terrorism, natural disaster, and all hazards. Game Theoretic Risk Analysis of Security Threats (Azaiez, editors, Springer, New York), 65–97.

Hwang, C., Tillman, F., Lee, M. (1981) System-reliability evaluation techniques for complex/large systems--A review. *IEEE Transactions on Reliability* R-30: 416–423.

Ileto, J. (2011) Improving resiliency of the petroleum supply chain for the Hawaiian Islands. Master's thesis, (Naval Postgraduate School, Monterey, California).

Long, C. (2013) Analyzing the resilience of the U.S. military fuel distribution system for mainland Japan. Master's thesis, (Naval Postgraduate School, Monterey, California).

Parnell, G., Liebe, R., Dillon-Merrill, R., Buede, D., Scouras, J., Colletti, B., Cummings, M., McGarvey, D., Newport, R., Vinch, P., Ayyub, B., Kaminskiy, M., Scouras, J. (2005) *Homeland Security risk assessment Volume I – An illustrative framework and Volume II- Appendices of methods*. (Homeland Security Institute, Washington, DC).

Paté-Cornell, M., Guikema, S. (2002) Probalistic modeling of terrorist threats: A systems analysis approach to setting priorities among countermeasures. *Mil Ops Res* **7**(4) 5–23.

National Aeronautics and Space Administration (NASA) (2011) Probabilistic Risk Assessment Procedures Guide for NASA Managers and Practitioners (Second Edition, Washington, DC).

Salmerón, J., Wood, K., Baldick R. (2004) Analysis of electric grid security under terrorist threat. *IEEE Transactions on Power Systems* 19 905–912.

Salmerón, J., Wood, K., Baldick R. (2009) Worst-case interdiction analysis of large-scale Electric power grids. *IEEE Transactions on Power Systems* 24(1) 96–104.

White House, The (2013) Presidential Policy Directive- Critical Infrastructure Security and Resilience. (Washington, DC).

White House, The (2014) Fiscal year 2014 Budget of the U.S. Government. Retrieved from: http://www.whitehouse.gov/sites/default/files/omb/budget/fy2014/assets/budget.pdf

Willis, H. H. (2007) Guiding resource allocations based on terrorism risk. *Risk analysis* 27(3) 597–606.

Zhuang, J., Bier, V.M. (2007) Balancing terrorism and natural disasters-defensive strategy with endogenous attacker effort. *Ops Res*, 55(5):976–991.

THIS PAGE INTENTIONALLY LEFT BLANK

INITIAL DISTRIBUTION LIST

1. Defense Technical Information Center
   Ft. Belvoir, Virginia

2. Dudley Knox Library
   Naval Postgraduate School
   Monterey, California