2022-06

RUNTIME ANALYSIS OF BENDERS DECOMPOSITION AND DUAL ILP ALGORITHMS AS APPLIED TO COMMON NETWORK INTERDICTION PROBLEMS

Trask, Timothy S., Jr.

Monterey, CA; Naval Postgraduate School

NAVAL POSTGRADUATE SCHOOL

MONTEREY, CALIFORNIA

THESIS

RUNTIME ANALYSIS OF BENDERS DECOMPOSITION AND DUAL ILP ALGORITHMS AS APPLIED TO COMMON NETWORK INTERDICTION PROBLEMS

by

Timothy S. Trask Jr.

June 2022

Thesis Advisor:	Emily M. Craparo
Second Reader:	W. Matthew Carlyle

THIS PAGE INTENTIONALLY LEFT BLANK

REPORT DOCUMENTATION PAGE			Form Approved OMB No. 0704-0188

1. AGENCY USE ONLY *(Leave blank)*	2. REPORT DATE June 2022	3. REPORT TYPE AND DATES COVERED Master's thesis

4. TITLE AND SUBTITLE
RUNTIME ANALYSIS OF BENDERS DECOMPOSITION AND DUAL ILP ALGORITHMS AS APPLIED TO COMMON NETWORK INTERDICTION PROBLEMS

5. FUNDING NUMBERS

6. AUTHOR(S) Timothy S. Trask Jr.

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.

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

12b. DISTRIBUTION CODE
A

13. ABSTRACT (maximum 200 words)

Attacker-defender models help practitioners understand a network's resistance to attack. An assailant interdicts a network, and the operator responds in such a way as to optimally utilize the degraded network. This thesis analyzes two network interdiction algorithms, Benders decomposition and a dual integer linear program approach, to compare their computational efficiency on the shortest path and maximum flow interdiction problems. We construct networks using two operationally meaningful structures: a grid structure designed to represent an urban transportation network, and a layered network designed to mimic a supply chain. We vary the size of the network and the attacker's budget and we record each algorithm's runtime.

Our results indicate that Benders decomposition performs best when solving the shortest path interdiction problem on a grid network, the dual integer linear program performs better for the maximum flow problem on both the grid and layered network, and the two approaches perform comparably when solving the shortest path interdiction problem on the layered network.

14. SUBJECT TERMS	15. NUMBER OF PAGES
network interdiction, Benders decomposition, integer linear program, ILP, dual ILP, integer linear program, runtime	55
	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.

RUNTIME ANALYSIS OF BENDERS DECOMPOSITION AND DUAL ILP ALGORITHMS AS APPLIED TO COMMON NETWORK INTERDICTION PROBLEMS

Timothy S. Trask, Jr.
Lieutenant Commander, United States Navy
BBA, University of North Florida, 2010

Submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE IN OPERATIONS RESEARCH

from the

NAVAL POSTGRADUATE SCHOOL
June 2022

Approved by: Emily M. Craparo
Advisor

W. Matthew Carlyle
Second Reader

W. Matthew Carlyle
Chair, Department of Operations Research

THIS PAGE INTENTIONALLY LEFT BLANK

ABSTRACT

Attacker-defender models help practitioners understand a network's resistance to attack. An assailant interdicts a network, and the operator responds in such a way as to optimally utilize the degraded network. This thesis analyzes two network interdiction algorithms, Benders decomposition and a dual integer linear program approach, to compare their computational efficiency on the shortest path and maximum flow interdiction problems. We construct networks using two operationally meaningful structures: a grid structure designed to represent an urban transportation network, and a layered network designed to mimic a supply chain. We vary the size of the network and the attacker's budget and we record each algorithm's runtime.

Our results indicate that Benders decomposition performs best when solving the shortest path interdiction problem on a grid network, the dual integer linear program performs better for the maximum flow problem on both the grid and layered network, and the two approaches perform comparably when solving the shortest path interdiction problem on the layered network.

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

Executive Summary

Whether it consists of information, commodities, or vehicular traffic, network flow continuity is critical for the functioning of many complex and important systems, particularly those revolving around health care, energy, logistics, transportation, and communication infrastructure. Identifying vulnerabilities in a timely manner is critical to the continued uninterrupted flow of these networks. Disrupting any of these networks for an appreciable amount of time has been shown to immediately and drastically impact many facets of life. For instance, Svitek (2021) reports power outages caused by winter storms claimed 246 lives in Texas in February of 2021, and INRIX (2019) shows that traffic congestion causes Americans to lose on average 97 hours per year at a cost of $1,348.00 per driver. To prevent such losses and disruptions, complex networks require constant analysis to identify vulnerabilities and determine where limited resources can best be invested to mitigate these vulnerabilities.

Practitioners identify and fortify against vulnerabilities in their systems by solving so-called "attacker-defender" models where a network is optimally attacked and defended resulting in an optimized attack and defender pattern that may aim to maximize throughput or identify vulnerabilities. As the networks grow in size and complexity, these problems can take significant computational energy and time to solve.

This thesis applies the Benders decomposition and a dual integer linear program approach to the shortest path and maximum flow problems. The network topologies studied include grid and layered networks, and vary in size and density. We record and analyze the impact of network characteristics and attacker's budget on model runtime to identify which algorithm is better for various problem setups.

Our analysis concludes that the Benders decomposition approach is generally better suited for expeditiously finding the shortest path on a grid network while the dual integer linear program approach is better suited for identifying the maximum flow on both the grid and layered style network. Neither algorithm conclusively performed better when finding the shortest path on a layered network.

References

INRIX (2019) Inrix: Congestion costs each American 97 hours, $1,348 a year. https://inrix.com/press-releases/scorecard-2018-us/.

Svitek P (2021) Texas puts final estimate of winter storm death toll at 246. https://www.texastribune.org/2022/01/02/texas-winter-storm-final-death-toll-246/.

Acknowledgments

My deepest gratitude goes to Professors Emily Craparo and Matthew Carlyle for their guidance, direction, and patience throughout the entire thesis process. They have jointly turned an otherwise arduous experience into a wonderful learning opportunity for myself and the field of optimization. I am also indebted to Professor Yael Grushka-Cockayne for sparking and fostering my initial interest in Operations Research. Without these great educators and researchers, none of this would have been possible for me. My love and gratitude go to my wife and daughters, Bethel, Maven, and Margot, without whose unwavering love and support this venture would have been impossible.

THIS PAGE INTENTIONALLY LEFT BLANK

CHAPTER 1:
Introduction

Whether it consists of information, commodities, or vehicular traffic, network flow continuity is critical for the functioning of many complex and important systems, particularly those revolving around health care, energy, logistics, transportation, and communication infrastructure. Identifying vulnerabilities in a timely manner is critical to the continued uninterrupted flow of these networks. Disrupting any of these networks for an appreciable amount of time has been shown to immediately and drastically impact many facets of life. For instance, Svitek (2021) reports power outages caused by winter storms in Texas claimed 246 lives in February of 2021, and INRIX (2019) reports that traffic congestion causes Americans to lose on average 97 hours per year at a cost of $1,348.00 per driver. To prevent such losses and disruptions, complex networks require constant analysis to identify vulnerabilities and determine where limited resources can best be invested to mitigate these vulnerabilities.

Practitioners identify and fortify against vulnerabilities in their systems by solving so-called "attacker-defender" models. Solving such models involves positing attacker capabilities, as well as the potential damage that could be inflicted on the network. Where these parameters are uncertain, it may be advantageous to consider many instances with varying input data. Moreover, networks of practical scale may be very large, containing hundreds of thousands of nodes and arcs. As such, computational efficiency is a key concern. Practitioners use a combination of theoretical (big-O) runtimes and practical experience to select algorithms for their studies. While runtime analysis and practical experience are useful, systematic empirical studies of the commonly used algorithms are lacking in the literature.

Various algorithms exist to identify where a network is vulnerable or where it should be fortified. Some of these algorithms may offer advantages in terms of computational efficiency, but the magnitude of the improvement likely depends on the exact form and magnitude of the problem being solved. Parameters for these problems primarily include the size and shape of the network and the attacker's budget.

This thesis performs a computational study of two common approaches to solving network

interdiction and design problems, namely Benders decomposition (BD) and a dual integer linear program (DILP) approach. We study the relative efficiency of these algorithms on two common network flow interdiction problems: the shortest path interdiction problem and the maximum flow interdiction problem.

By comparing BD and DILP on a range of problem types, we hope to provide insight into the relative efficiency of these algorithms, and how their efficiency varies across problem types and size. With this insight, operators can provide timely recommendations on where to invest resources to mitigate, redirect, or respond to damaged networks.

1.1 Background

Network applications vary widely and include tasks both mundane and extraordinary. Heisler (2015) identifies that both Google and Apple use Dijkstra's algorithm to find the shortest path to the grocer. (Wood (1993) targets the flow of tracer chemicals can reduce throughput in drug trafficking networks. Bartolacci and Dimitrov (2017) identifies improvements in a cellular network can improve critical message dissemination during national emergencies. Ishimatsu et al. (2016) analyzes multicommodity flow networks identifies where resources should be located to lower costs, optimize sustainability, and maximize throughput.

1.1.1 Shortest Path and Maximum Flow Problems

We consider two classical network flow problems, both of which are defined on a network consisting of nodes in set N (indexed by i and j) and arcs (i, j) in set A. In the s-t shortest path problem, a cost $c_{i,j}$ is associated with each arc $(i, j) \in A$, and the goal is to send a unit of flow along a directed path from start node s to terminal node t at minimum total cost. Decision variables $X_{i,j}$ indicate the flow along arc $(i, j) \in A$. The mathematical representation of the s-t shortest path problem is presented as follows.

$$
\begin{aligned}
\min_{X} \quad & \sum_{(i,j)\in A} c_{i,j} X_{i,j} \\
\text{s.t.} \quad & \sum_{j:(i,j)\in A} X_{i,j} - \sum_{j:(j,i)\in A} X_{j,i} = \begin{cases} 1 & i = s \\ 0 & i \neq s,t \\ -1 & i = t \end{cases} \quad \forall i \in N \\
& X_{i,j} \geq 0 \qquad \forall (i,j) \in A
\end{aligned}
$$

In the s-t maximum flow problem, each arc $(i, j) \in A$ has a capacity $u_{i,j}$ associated with it, representing the maximum amount of flow that can travel along that arc. The goal is to maximize the total flow from start node s to terminal node t, where again decision variables $X_{i,j}$ represent the flow along each arc $(i, j) \in A$. We utilize a "virtual arc" (t, s) (not included in A) to connect the terminal node back to the source node, allowing flow to circulate through the network. We assume all capacities are finite, and define $U \equiv \max_{(i,j)\in A}\{u_{i,j}\}$ as the maximum capacity on any single arc in the network. The overall flow in the network, and therefore the flow on the artificial return arc, (t, s), is then bounded by $|N|U$.

$$
\begin{aligned}
\max_{X} \quad & X_{t,s} \\
\text{s.t.} \quad & \sum_{j:(i,j)\in A} X_{i,j} - \sum_{j:(j,i)\in A} X_{j,i} = \begin{cases} X_{t,s} & i = s \\ 0 & i \neq s,t \\ -X_{t,s} & i = t \end{cases} \quad \forall i \in N \\
& 0 \leq X_{i,j} \leq u_{i,j} \qquad \forall (i,j) \in A \\
& 0 \leq X_{t,s} \leq |N|U
\end{aligned}
$$

1.1.2 Attacker-Defender Models for Shortest Path and Maximum Flow

For both the shortest path and maximum flow problem, it is possible to formulate an attacker-defender model in which an attacker wishes to expend a finite set of resources as efficiently

as possible, with the goal of producing the worst possible optimal objective value for an operator of the respective network.

In the shortest path interdiction problem, the attacker must choose a set of at most *max_attacks* arcs to attack, where an attacked arc (i, j) has its cost increased by penalty value $q_{i,j}$ (in addition to its original cost $c_{i,j}$). Letting binary decision variables $Y_{i,j}$ represent the decision of whether or not to attack each arc $(i, j) \in A$, the attacker's problem is presented as follows.

$$\max_{Y} \min_{X} \quad \sum_{(i,j)\in A} (c_{i,j} + q_{i,j} Y_{i,j}) X_{i,j}$$

$$s.t. \quad \sum_{j:(i,j)\in A} X_{i,j} - \sum_{j:(j,i)\in A} X_{j,i} = \begin{cases} 1 & i = s \\ 0 & i \neq s, t \\ -1 & i = t \end{cases} \qquad \forall i \in N$$

$$\sum_{(i,j)\in A} Y_{i,j} \leq max\_attacks$$

$$Y_{i,j} \in \{0, 1\} \qquad \forall (i, j) \in A$$

$$X_{i,j} \geq 0 \qquad \forall (i, j) \in A$$

Of note, the attacker's and operator's decision variables interact only in the objective function, where the total cost of the path chosen by the operator is a function of both the attacker's and the operator's decisions.

Similarly, the maximum flow interdiction problem represents the impact of the attacker's decisions via the objective function, where any flow that is sent along attacked arcs is "canceled out" of the objective function using a penalty $q_{i,j} > 1$.

$$\min_{Y} \max_{X} \quad X_{t,s} - \sum_{(i,j)\in A} q_{i,j} Y_{i,j} X_{i,j}$$

$$s.t. \quad \sum_{j:(i,j)\in A} X_{i,j} - \sum_{j:(j,i)\in A} X_{j,i} = \begin{cases} X_{t,s} & i = s \\ 0 & i \neq s,t \\ -X_{t,s} & i = t \end{cases} \qquad \forall i \in N$$

$$\sum_{(i,j)\in A} Y_{i,j} \leq max\_attacks$$

$$Y_{i,j} \in \{0,1\} \qquad \forall (i,j) \in A$$

$$0 \leq X_{i,j} \leq u_{i,j} \qquad \forall (i,j) \in A$$

$$0 \leq X_{t,s} \leq |N|U$$

Both the shortest path and maximum flow interdiction models can be modified or embellished to suit particular applications. For instance, both models easily accommodate more sophisticated attack budget constraints than the simple cardinality constraints presented in the formulations above. Interdiction models can be deterministic, meaning that arc costs and capacities are known and unchanging and attacks always succeed, or they can be stochastic, meaning that the probability of successful interdiction and the arc capacities are uncertain. For the purposes of this thesis, all parameter values are deterministic and are known by both the operator/defender and the interdictor/attacker.

1.2 Literature Review

Ford and Fulkerson (1962) proposed the max-flow min-cut theorem, which states that "the maximum flow value from s to t is equal to the minimal cut capacity over cuts separating s and t" (p. 19).

Wollmer (1963) develops an algorithm that enumerates all arcs in a capacitated network to find the maximum flow. Lubore et al. (1971) improves on Willmer's algorithm by determining the single most vital arc with respect to maximum flow. Their algorithm assumes the interdiction costs for arcs are equal across the network and is applicable when determining the single most valuable arc but is not capable of finding the set of $k > 1$ most vital arcs. Later, Lubore et al. (1975) developed an algorithm that minimizes the maximum flow of the network by finding a set of k arcs that, when removed decreased the network's throughput and is applicable to both planar and nonplanar networks.

Wollmer (1964) focuses on sensitivity analysis of the flow across a planar network by removing n number of arcs from the network in order to minimize flow. By a modification to the location of the capacities from nodes to arcs, the problem can be viewed as a shortest path or lowest cost issue.

Wollmer (1970) outlines two algorithms specifically addressing network interdiction that target both the maximum flow and minimum cost flow problems. Helmbold (1971) proposes the use of dynamic programming to solve a generalized version of Wollmer's model in which he uses recursion to find the shortest path across the network. Preston (1972) identifies optimal allocation of aircraft for airstrikes against a transportation network and the added benefit of assigning another aircraft to interdict the network.

Golden and Ball (1978) develops a model that lengthens the arcs in a network via a least cost investment strategy to increase the cost of the shortest path. While the model identifies the minimum cost set of attacks that will increase the shortest path, it does not allow for arcs to be removed from the network.

Van Roy (1983) proposes the use of cross decomposition utilizing both the primal and dual structures of the problem. Holmberg (1990) further explores the tractability of cross decomposition as it relates to specifically linear and generally non-linear problems. The first of these two models requires that part of the problem be linear and results in finite

convergence. The second problem deals with entirely nonlinear problems.

Cormican et al. (1998) advocates splitting difficult to solve problems into several easier to solve ones and explores implications of uncertainty in network interdiction.

Fischetti et al. (2010) propose new criterion for determining Benders cuts specifically targeting unbounded subproblems that are significantly more robust and often yield faster results than the standard Benders cuts. Similarly, Wood (2002) seeks to analyze Benders cuts as they apply to the *s*-*t* shortest path problem. The first algorithm adds "supervalid inequalities" to the master problem which eliminate the most recent solution but are guaranteed not to eliminate an optimal solution. The second algorithm institutes a covering strategy in which at least one arc of a subpath must be interdicted. Once no more subpaths can be interdicted, an optimal solution has been found.

Steinrauf (1991) develops two algorithms. The first identifies a set of arcs to interdict that will minimize flow. The second seeks to interdict a set of arcs that seeks to maximize the number of nodes isolated based on a finite interdiction budget.

Wood (1993) address how attack resources are spent on breaking arcs. The first constrains the problem by requiring that an arc must be attacked to the point at which it breaks. The second requires that only one unit of resources is required to break an arc so a cardinality constraint is put on the attack budget dictating how many arcs can be attacked.

Attacker-defender models have been used in identifying network vulnerability. Garcia (2001) develops a process for assessing the vulnerability of a network based on random disruption through analysis of its individual components as they relate to the whole system.

Brown et al. (2006) develops bi-level and tri-level defender-attacker-defender models that assume an intentional intelligent attacker and defender and seek to develop worst-case scenario models and identify the "value of protecting or hardening" a subset of the network. (p. 531). Their models use attacker-defender models exactly like the ones we analyze as subproblems. Being able to solve those subproblems quickly leads to direct improvements in the run times of their algorithms.

Ross (2014) develops a model balancing worst-case and random disruptions but was only effective at smaller networks due to the computational intensity of the model. All of these

models used various versions of a bi-level attacker-defender model or tri-level defender-attacker-defender models.

Each of these approaches to attacker-defender models is critical in identifying how to best defend infrastructure.

CHAPTER 2:
Attacker-Defender Models

This chapter describes our methodology for generating instances of the interdiction problems described in Section 1.1.2, as well as the algorithms we use to solve them. We consider two network topologies: a grid network, which is designed to mimic an urban transportation network, and a layered network, which is designed to represent a supply chain. For each problem type and network topology, we compare the performance of the BD algorithm and the DILP on many randomly-generated networks of varying sizes. We are primarily interested in applying an attacker defender model to basic network problems to determine which of two algorithms will return an optimal solution in the least amount of time. We will do this many times on many different networks to determine which algorithm is best suited for each problem.

This chapter presents how we generate the networks, gives a mathematical formulation of the interdiction problems discussed in Chapter 1, and discusses the algorithms used to determine an optimal attack and the corresponding operator solution.

2.1 Network Development

We now describe our methodology for generating network instances, including the set of nodes N present in each instance, the set of arcs A that connect these nodes, and any applicable data, including arc costs $c_{i,j}$ for the shortest path problem, arc capacities $u_{i,j}$ for the maximum flow problem, and penalties $q_{i,j}$ for both problems.

2.1.1 Grid Network

Our grid network instances connect s to t via a grid of nodes with an equal number of nodes in height and width, as shown by nodes 1-9 in Figure 2.1. We refer to this number as the *grid dimension*. As an example, if the grid dimension was defined as 5, that would result in a 5x5 grid containing 25 nodes, for a total of 27 nodes in the network (including s and t).

The s node has arcs that flow into every node on the left side of the grid, and each node

on the right side of the grid has an arc that flows into node t. Within the grid, arcs connect each node to those nodes immediately above, below, left, and right of it. For each diagonal pair of nodes (i, j), such as nodes $i = 4$ and $j = 8$ in Figure 2.1, arc (i, j) appears with 20% probability. This is denoted by the dashed lines in Figure 2.1.

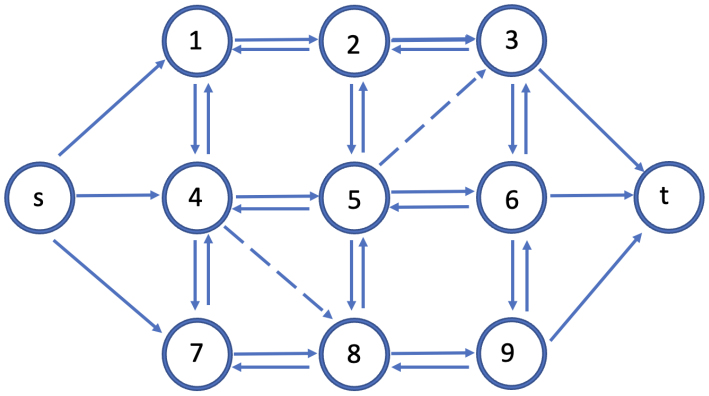


Figure 2.1: Example: Grid network with dimension 3.

For each arc (i, j), we generate a cost $c_{i,j}$, shortest path penalty $q_{i,j}$, and capacity $u_{i,j}$ value uniformly at random between specified lower and upper limits. For the purposes of this thesis, the lower and upper limits for the grid network parameters are provided in Table 2.1.

	Lower Limit	Upper Limit
Cost $c_{i,j}$	10	20
Penalty $q_{i,j}$	30	40
Capacity $u_{i,j}$	5	15

Table 2.1: Arc parameters: Grid network

For the maximum flow interdiction problem, any $q_{i,j} > 1$ suffices. This thesis uses a value of $q_{i,j} = 1.05$ for all $(i, j) \in A$.

2.1.2 Layered Network

The layered network is characterized by the number of layers and the number of nodes per layer. Our layered network instances connect s to t via a series of layers containing a fixed number of nodes n. Each node in layer l is connected via an arc (i, j) to a random number of nodes $connectedness$ in layer $l + 1$. For each node, the value of $connectedness$ is chosen uniformly at random between specified lower and upper limits. Additionally, for each node i in layer l, there is a 10% chance that an arc exiting i will "skip" a layer and connect i to some node j in layer $l + 2$. For example, in Figure 2.2 we have Three layers with five nodes in each layer, and each node i is connected to at least 1 but not more than 3 nodes j in the adjacent layer (so $1 \leq connectedness \leq 3$). The arc $(1, 11)$ represents an arc that skips a layer; this is indicated by the dashed arc line.

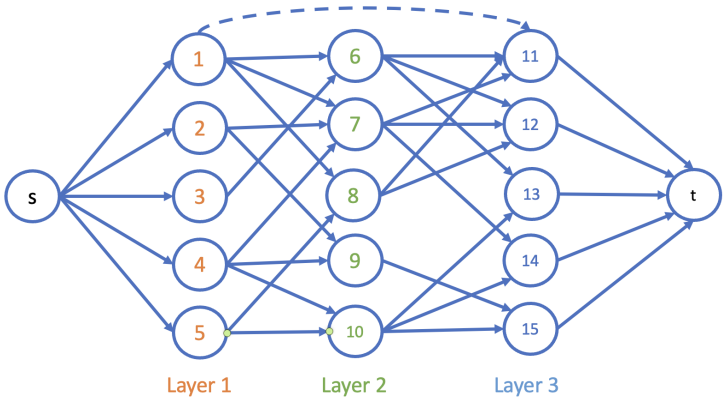


Figure 2.2: Example: Three-layer network with five nodes in each layer.

As in the grid network, each arc $(i, j) \in A$ is assigned a cost $c_{i,j}$, shortest path penalty $q_{i,j}$, and capacity $u_{i,j}$ uniformly at random between specified lower and upper limits. Again, we set $q_{i,j} = 1.05$ for all arcs in the maximum flow problem. Each layer contains $n = 15$ nodes. The lower and upper limits on the $connectedness$, $c_{i,j}$, $q_{i,j}$, and $u_{i,j}$ parameters are provided in Table 2.2.

To ensure that the network is connected, a breadth-first search is conducted starting and ending at nodes s and t, where a reverse adjacency list is used when starting from node t. The search marks each node the first time it is encountered. Once the search is concluded, the list of marked nodes is compared to the master list of nodes. If a node is not in the

marked node list, the network is discarded and regenerated until a connected network is generated.

Table 2.2: Arc parameters: Layered network

	Lower Limit	Upper Limit
$connectedness$	2	6
Cost $c_{i,j}$	10	20
Penalty $q_{i,j}$	30	40
Capacity $u_{i,j}$	5	15

2.2 Algorithms for Solving Attacker-Defender Problems

We compare two methods for solving the attacker-defender problems described in Section 1.1.2: the DILP approach and BD. We now describe those methods in detail.

2.2.1 Dual Integer Linear Program Approach

Our first solution technique, DILP, is based on the observation that for a fixed attack Y, the operator's problem is a linear program that could equivalently be solved via its dual.

Shortest Path

To develop our dual integer linear program, we associate dual variables π with the operator's main constraints in the *s*-*t* shortest path interdiction problem.

$$\max_{Y} \min_{X} \quad \sum_{(i,j)\in A} (c_{i,j} + q_{i,j} Y_{i,j}) X_{i,j}$$

$$s.t. \quad \sum_{j:(i,j)\in A} X_{i,j} - \sum_{j:(j,i)\in A} X_{j,i} = \begin{cases} 1 & i = s \\ 0 & i \neq s, t \\ -1 & i = t \end{cases} \qquad \forall i \in N \qquad [\pi_i]$$

$$\sum_{(i,j)\in A} Y_{i,j} \leq max\_attacks$$

$$Y_{i,j} \in \{0, 1\} \qquad \forall (i, j) \in A$$

$$X_{i,j} \geq 0 \qquad \forall (i, j) \in A$$

Taking the dual of the inner minimization linear program, we obtain the following maxi-mization problem.

$$\max_{\pi, Y} \quad \pi_s - \pi_t$$

$$s.t. \quad \pi_i - \pi_j - q_{i,j} Y_{i,j} \leq c_{i,j} \qquad \forall (i, j) \in A$$

$$\sum_{(i,j)\in A} Y_{i,j} \leq max\_attacks$$

$$Y_{i,j} \in \{0, 1\} \qquad \forall (i, j) \in A$$

$$\pi_i \quad \text{unrestricted} \qquad \forall i \in N$$

$$\pi_s \equiv 0$$

Our DILP approach involves simply solving this ILP to an optimality gap of 1%.

Maximum Flow

Proceeding as with the *s-t* shortest path problem, we introduce dual variables π and α associated with the operator's constraints.

$$\min_{Y} \max_{X} \quad X_{t,s} - \sum_{(i,j)\in A} q_{i,j} Y_{i,j} X_{i,j}$$

$$s.t. \quad \sum_{j:(i,j)\in A} X_{i,j} - \sum_{j:(j,i)\in A} X_{j,i} = \begin{cases} X_{t,s} & i = s \\ 0 & i \neq s,t \\ -X_{t,s} & i = t \end{cases} \qquad \forall i \in N \qquad [\pi_i]$$

$$\sum_{(i,j)\in A} Y_{i,j} \leq max\_attacks$$

$$Y_{i,j} \in \{0,1\} \qquad \forall (i,j) \in A$$

$$0 \leq X_{i,j} \leq u_{i,j} \qquad \forall (i,j) \in A \qquad [\alpha_{i,j}]$$

$$0 \leq X_{t,s} \leq |N|U \qquad [\alpha_{t,s}]$$

Taking the dual of the inner maximization linear program, we obtain the following minimization problem.

$$\min_{\pi,\alpha,Y} \quad \sum_{(i,j)\in A} u_{i,j}\alpha_{i,j} + |N|U\alpha_{t,s}$$

$$\text{s.t.} \quad \pi_i - \pi_j + \alpha_{i,j} + q_{i,j}Y_{i,j} \geq 0$$

$$\pi_t - \pi_s + \alpha_{t,s} \geq 1 \qquad \forall (i,j) \in A$$

$$\pi_i \quad \text{unrestricted} \qquad \forall i \in N$$

$$\pi_s \equiv 0$$

$$\alpha_{i,j} \geq 0 \qquad \forall (i,j) \in A$$

$$\alpha_{t,s} \geq 0$$

$$\sum_{(i,j)\in A} Y_{i,j} \leq max\_attacks$$

$$Y_{i,j} \in \{0,1\} \qquad \forall (i,j) \in A$$

As with the shortest path DILP, we solve this ILP to an optimality gap of 1%.

2.2.2 Benders Decomposition Approach

Our second approach, BD, solves the attacker's problem by iterating between an operator subproblem, which evaluates the effect of a given attack, and the attacker's relaxed master problem, which identifies the most promising feasible attack, given a set of observed operator solutions.

Shortest Path

For the shortest path problem, the Benders subproblem finds the operator's shortest path, given a set of attacked arcs, while the master problem finds the best feasible attack, given the set of operator paths observed so far.

Shortest Path Operator's Subproblem

The shortest path operator's subproblem is as follows.

$$\min_{X} \quad \sum_{(i,j)\in A} (c_{i,j} + q_{i,j}\hat{Y}_{i,j})X_{i,j}$$

$$s.t. \quad \sum_{j:(i,j)\in A} X_{i,j} - \sum_{j:(j,i)\in A} X_{j,i} = \begin{cases} 1 & i = s \\ 0 & i \neq s,t \\ -1 & i = t \end{cases} \qquad \forall i \in N$$

$$X_{i,j} \geq 0 \qquad \forall (i,j) \in A$$

The subproblem fixes the values for $\hat{Y}_{i,j}$ based on a feasible attack solution and provides a lower bound on the attacker's optimal objective value. The subproblem also provides a new operator extreme point (i.e., a path) that is used in the master problem.

Shortest Path Attacker's Relaxed Master Problem

The attacker's relaxed master problem finds the best feasible attack against a set of operator extreme points $\hat{X}_{i,j}^k$, for $k = 1, ..., K$, while variable Z records the objective value that could be achieved by the attacker if the operator were restricted to the given extreme points.

$$\begin{aligned}
\max_{Y,Z} \quad & Z \\
s.t. \quad & Z \leq \sum_{(i,j)\in A} (c_{i,j} + q_{i,j}) \hat{X}_{i,j}^k Y_{i,j} && k = 1, 2, ..., K \\
& \sum_{(i,j)\in A} Y_{i,j} \leq max_attacks \\
& Y_{i,j} \in \{0, 1\} \qquad \forall (i,j) \in A
\end{aligned}$$

The solution to this master problem provides a new attack plan to be utilized in the subproblem. It also provides an upper bound on the attacker's optimal objective value.

Benders Decomposition Algorithm

Given the subproblem and master problem formulations, the Benders decomposition algorithm for shortest path interdiction is:

1. $LB = -\infty; UB = \infty; K = 1$
2. $\hat{Y} = \mathbf{0}$ (i.e., $\hat{Y}_{i,j} = 0 \ \ \forall (i,j) \in A$)
3. While $UB - LB > \varepsilon LB$:
4. Solve OPERATOR SUBPROBLEM (using $\hat{Y}$) to get $\hat{X}^K = X^*, Q^K = \sum_{(i,j)\in A} \left(c_{i,j} + q_{i,j}\hat{Y}_{i,j}^k\right) X_{i,j}^*$
5. If $LB < Q^K$: $LB = Q^K, \hat{Y}^{BEST} = \hat{Y}$
6. Solve ATTACKER RELAXED MASTER (using $\hat{X}^1, \hat{X}^2, \ldots \hat{X}^K$) to get $\hat{Y} = Y^*, Z^K = Z^*$
7. If $UB > Z^K$: $UB = Z^K$
8. $K = K + 1$
9. End While
10. Return $\hat{Y}^{BEST}$ as an ε-optimal solution to the attacker problem

11. (optional) Solve OPERATOR SUBPROBLEM (using $\hat{Y}^{BEST}$) to determine operator's optimal response.

In our experiments, we solve to a tolerance of $\varepsilon = 1\%$.

Maximum Flow

For the maximum flow problem, the Benders subproblem finds the operator's maximum flow, given a set of attacked arcs, while the master problem finds the best feasible attack, given the set of operator flows observed so far.

Maximum Flow Operator's Subproblem

The maximum flow operator's subproblem is as follows.

$$\max_{X} \quad X_{t,s} - \sum_{(i,j)\in A} q_{i,j}\hat{Y}_{i,j}X_{i,j}$$

$$s.t. \quad \sum_{j:(i,j)\in A} X_{i,j} - \sum_{j:(j,i)\in A} X_{j,i} = \begin{cases} X_{t,s} & i = s \\ 0 & i \neq s,t \\ -X_{t,s} & i = t \end{cases} \qquad \forall i \in N$$

$$0 \leq X_{i,j} \leq u_{i,j} \qquad \forall (i,j) \in A$$

$$0 \leq X_{t,s} \leq |N|U$$

The subproblem fixes the values for $\hat{Y}_{i,j}$ based on a feasible attack solution and provides an upper bound on the attacker's optimal objective value. The subproblem also provides a new operator extreme point (i.e., a flow) that is used in the master problem.

Attacker's Relaxed Master Problem

The attacker's relaxed master problem finds the best feasible attack against a set of operator extreme points $\hat{X}^k_{i,j}$, for $k = 1, ..., K$, where dummy variable Z records the objective value that could be achieved by the attacker if the operator were restricted to the given extreme points.

$$
\begin{aligned}
\min_{Y,Z} \quad & Z \\
s.t. \quad & Z \geq \hat{X}^k_{t,s} - \sum_{(i,j)\in A} q_{i,j}\hat{X}^k_{i,j}Y_{i,j} && k = 1, 2, ..., K \\
& \sum_{(i,j)\in A} Y_{i,j} \leq max\_attacks \\
& Y_{i,j} \in \{0,1\} \qquad \forall (i,j) \in A
\end{aligned}
$$

The solution to this master problem provides a new attack plan to be utilized in the subproblem. It also provides a lower bound on the attacker's optimal objective value.

Benders Decomposition Algorithm

Given the subproblem and master problem formulations, the Benders decomposition algorithm for shortest path interdiction is:

1. $LB = -\infty; UB = \infty; K = 1$
2. $\hat{Y} = \mathbf{0}$ (i.e., $\hat{Y}_{i,j} = 0 \ \ \forall\, (i,j) \in A$)
3. While $UB - LB > \varepsilon LB$:
4. Solve OPERATOR SUBPROBLEM (using $\hat{Y}$) to get $\hat{X}^K = X^*, Q^K = X^*_{t,s} - \sum_{(i,j)\in A} q_{i,j}\hat{Y}_{i,j}X^*_{i,j}$
5. If $UB > Q^K$: $UB = Q^K$, $\hat{Y}^{BEST} = \hat{Y}$
6. Solve ATTACKER RELAXED MASTER (using $\hat{X}^1, \hat{X}^2, \ldots \hat{X}^K$) to get $\hat{Y} = Y^*, Z^K = Z^*$
7. If $LB < Z^K$: $LB = Z^K$
8. $K = K + 1$
9. End While
10. Return $\hat{Y}^{BEST}$ as an ε-optimal solution to the attacker problem
11. (optional) Solve OPERATOR SUBPROBLEM (using $\hat{Y}^{BEST}$) to determine operator's optimal response.

As with the shortest path problem, we solve to a tolerance of $\varepsilon = 1\%$.

CHAPTER 3:
Results

3.1 Hardware and Software

Experiments were run and figures were generated using an M1 Macbook Air (M1, 2020). This computer utilizes a 3.2 GHz 8-core central processor, a 1.28 GHz 7-core integrated graphics processor, and 8GB of unified LPDDR4X SDRAM. To the maximum extent practical, the computer was located in a cooled stationary location while conducting experiments.

MacOS Monterey (12.3) was used for all experiments. All experiments were conducted using Python 3.9 through the Spyder GUI. The solver utilized is CPLEX Studio 22.1.0. JMP Pro 16 was used to generate all figures.

All figures present the full results of our computational experiments, including the computation time of every instance run, as a function of problem size as quantified by the number of nodes in the network. In all tests we ran 15 replications for each size of network, for each type of network, at each parameter value. Gray dots on the figures represent actual results from these replications. Additionally, we use a cubic smoothing spline to smooth our results in all figures. Wood (2017) showed that the cubic smoothing spline balances the fit of the trend line to the data while generating a smooth trend line function.

A log transformation was applied to all figures along the *Runtime* axis to better view variation within data.

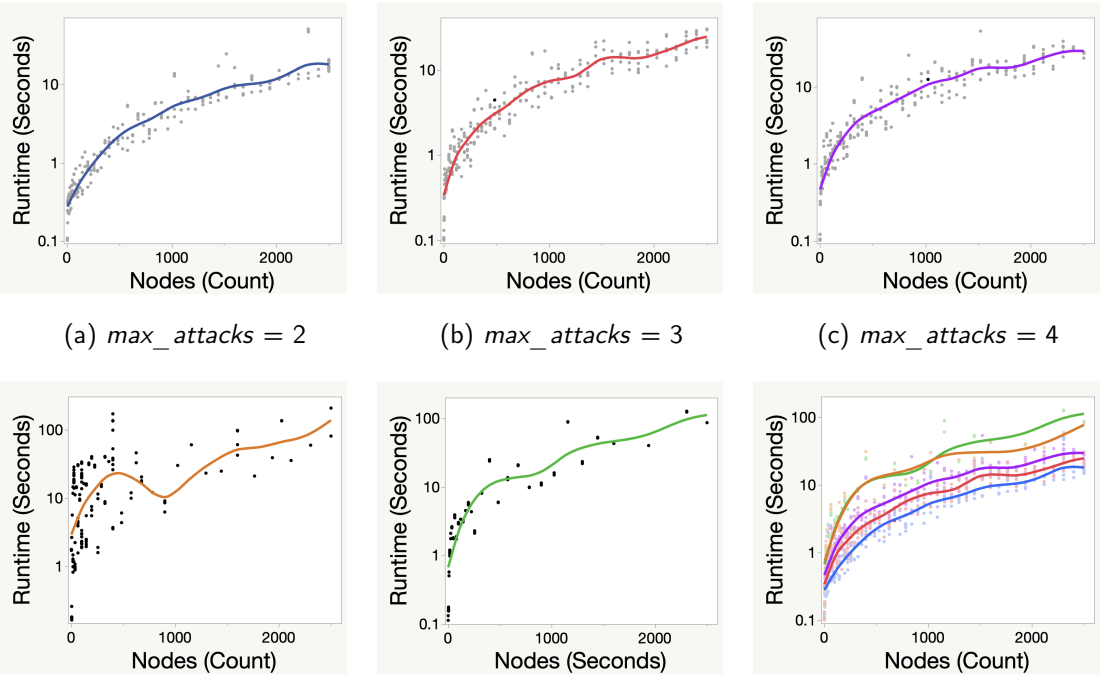
3.2 Shortest Path – Grid Network

Figures 3.1, 3.2, and 3.3 show runtime comparisons for the shortest path interdiction problem applied to a grid style network. The increase in nodes $|N|$ directly translates to an increase in network size.

3.2.1 BD

Figure 3.1 shows BD applied to the shortest path interdiction problem on a grid network at various network sizes and *max_attacks* values.

Our observations indicate that, as network size increases (i.e., as the number of nodes $|N|$ increases), the runtime of BD also increases.Figures 3.1 (d) and (e) show that at *max_attacks* values of 5 and 6, there is higher variation in smaller networks leading to an uncharacteristic decrease in runtime. Figure 3.1(f) shows that as *max_attacks* increases, the average runtime also increases at all observed network sizes.



(a) *max_attacks* = 2 (b) *max_attacks* = 3 (c) *max_attacks* = 4

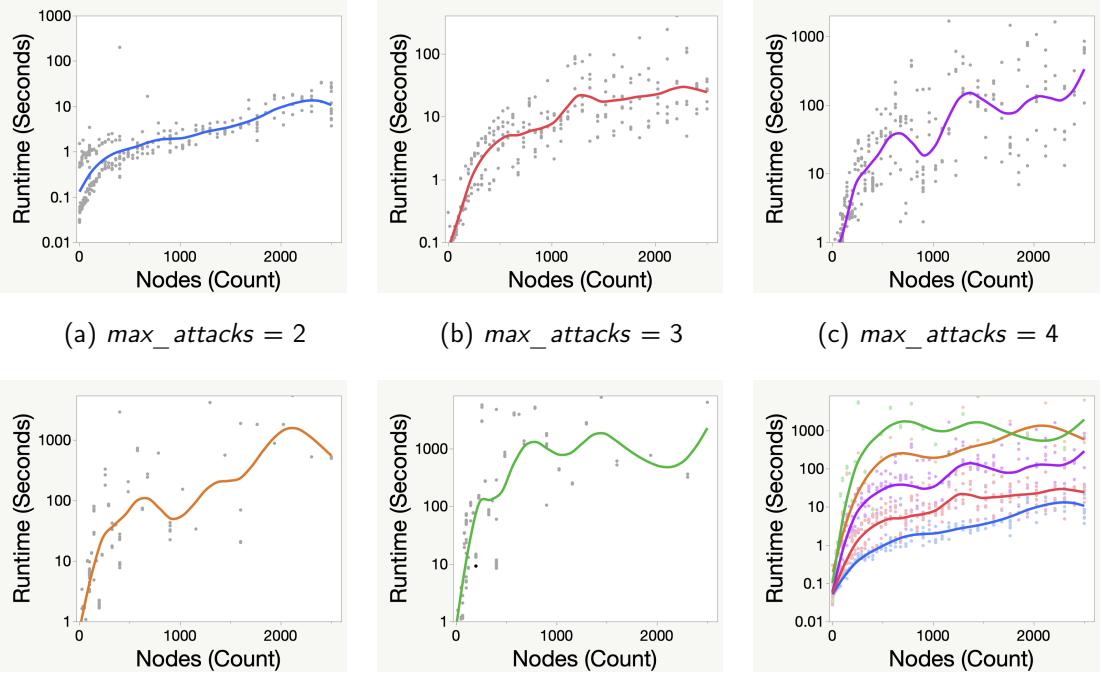
(d) *max_attacks* = 5 (e) *max_attacks* = 6 (f) *max_attacks* comparison

Figure 3.1: Runtime comparison: BD shortest path on a grid network.

3.2.2 DILP

Figure 3.2 shows DILP applied to the shortest path problem on a grid network at various network sizes and *max_attacks* values.

Our observations indicate that, a positive direct relationship exists between DILP runtime and network size. This is similar to how BD behaves. Figure 3.2(a) shows low relative variability among data points while Figures 3.2(b) through (e) show increasing variability and can be seen very clearly when comparing different *max_attacks* levels (see Figure 3.2(f)).



(a) *max_attacks* = 2 (b) *max_attacks* = 3 (c) *max_attacks* = 4

(d) *max_attacks* = 5 (e) *max_attacks* = 6 (f) *max_attacks* comparison

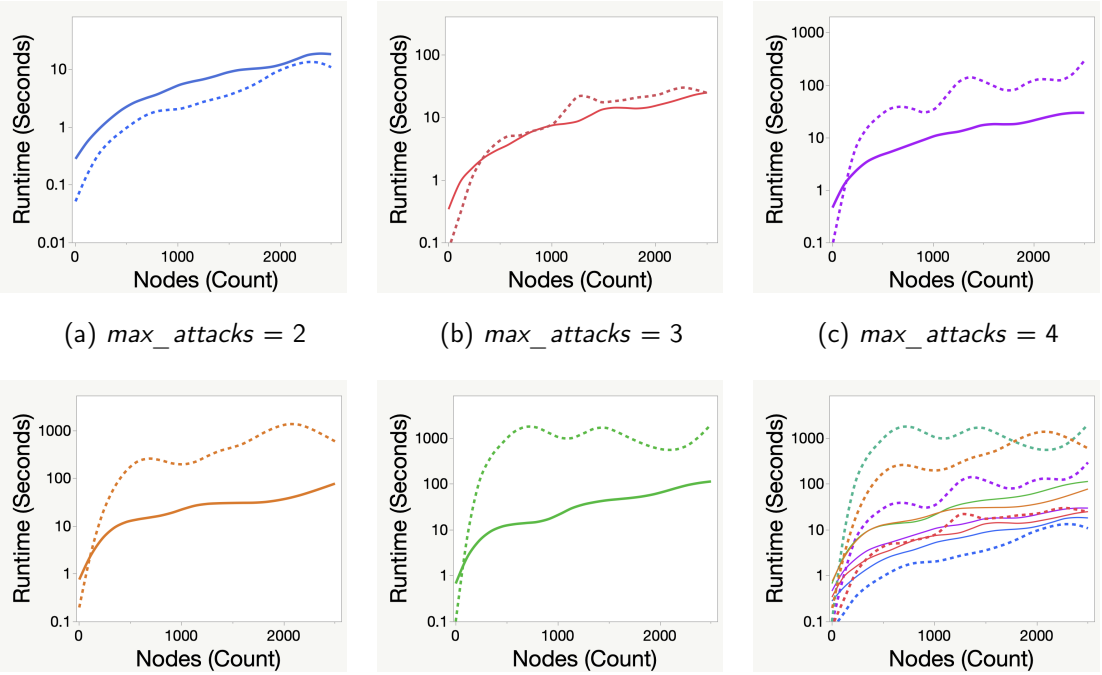
Figure 3.2: Runtime comparison: DILP shortest path on a grid network.

3.2.3 BD vs. DILP

Figures 3.3 shows a comparison of BD and DILP runtimes when applied to the shortest path problem on a grid network at various network sizes and *max_attacks* values.

In our observations, at *max_attacks* = 2, DILP returns an optimized solution quicker than BD. The two algorithms perform similarly for *max_attacks* = 3. For *max_attacks* > 3 , BD consistently returns a lower runtimes. At *max_attacks* > 3, as *max_attacks* continues to increase, the relative gap between BD and DILP runtimes increases (see Figures 3.3 (c)-(e)).

Figure 3.3(c) demonstrates the relative volatility in runtimes between BD and DILP on this particular problem. Figure 3.3(d) shows that at *max_attacks* = 5 the same relative drop in runtimes is seen in both BD and DILP at network sizes ranging from approximately 450 to 1,000 nodes before resuming an upward trend. At *max_attacks* = 6 (Figure 3.3(e)) this relationship is not as apparent.



(a) *max_attacks* = 2 (b) *max_attacks* = 3 (c) *max_attacks* = 4

(d) *max_attacks* = 5 (e) *max_attacks* = 6 (f) *max_attacks* comparison

Figure 3.3: Runtime comparison of BD (—) and DILP (- - -) for the shortest path interdiction problem on a grid network.

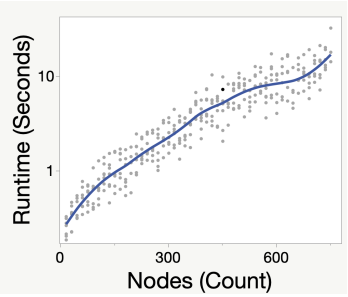
3.3 Shortest Path – Layered Network

Figures 3.4, 3.5, and 3.6 show runtime comparisons for the shortest path interdiction problem applied to a layered network at various network sizes and *max_attacks* values. The size of each layer was fixed at 15 nodes, so the increase in nodes corresponds to an increase in layers and overall size of the network.

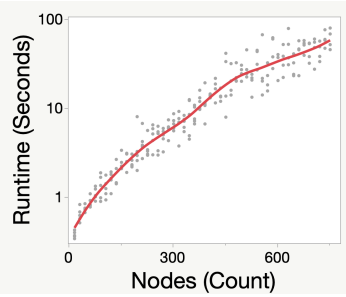
BD

Figure 3.4 shows runtimes when applying BD to the shorteset path problem on a layered networks at various network sizes and *max_attacks* values.

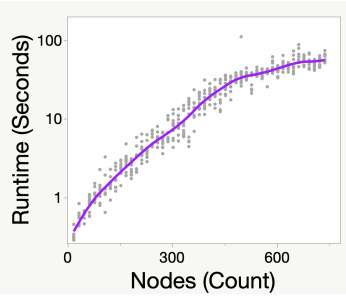
Our observations indicate that there is a positive relationship between network size and BD runtime. A comparison of each trend line across tested *max_attacks* levels indicates that, uniformly, as the *max_attacks* increases, so does the runtime. Variation among runtimes appears to be consistent across all *max_attacks* levels and network sizes.



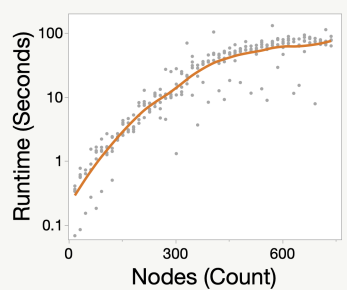
(a) *max_attacks* = 2



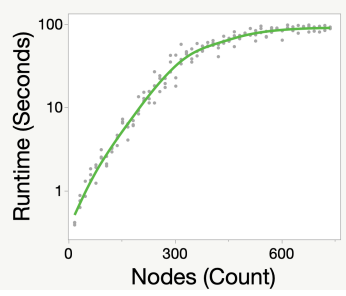
(b) *max_attacks* = 3



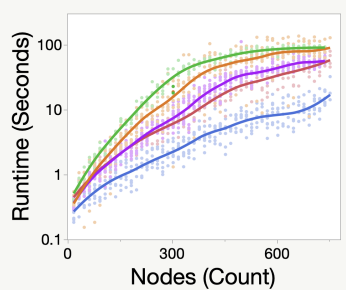
(c) *max_attacks* = 4



(d) *max_attacks* = 5



(e) *max_attacks* = 6



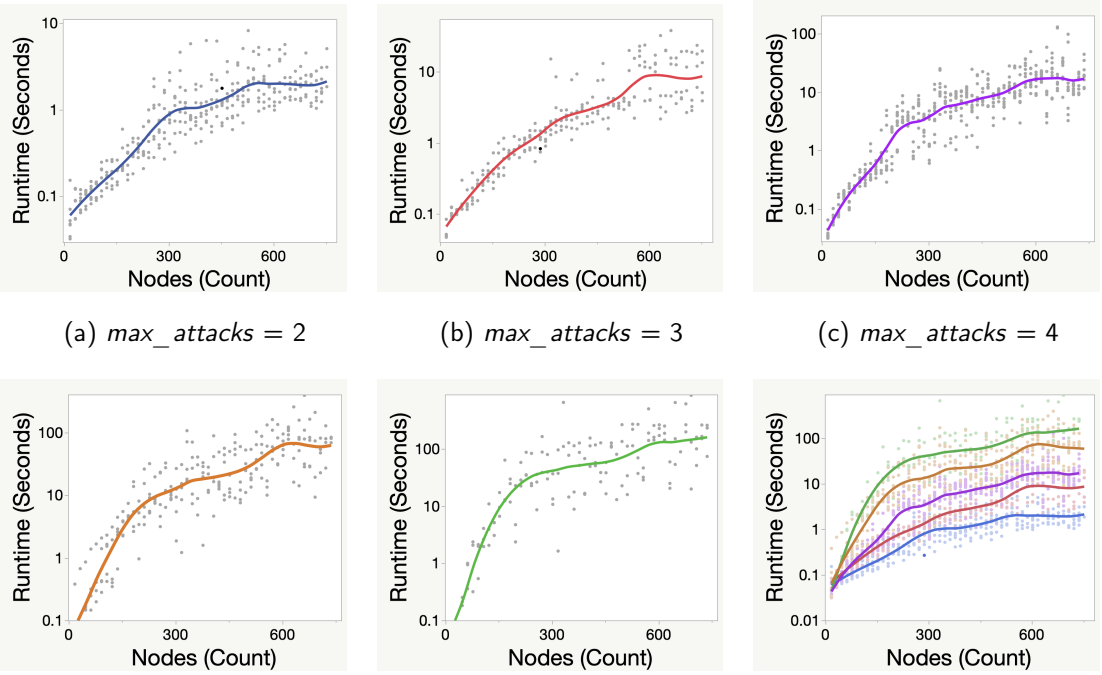
(f) *max_attacks* comparison

Figure 3.4: Runtime comparison: BD shortest path on a layered network.

DILP

Figure 3.5 shows runtimes when applying DILP to the shortest path problem on a layered networks at various network sizes and $max\_attacks$ values.

Our observations indicate that DILP runtime increases both as the network size increases and as *max_attacks* increases. The variation among runtimes also appears to increase as *max_attacks* increases.



(a) *max_attacks* = 2 (b) *max_attacks* = 3 (c) *max_attacks* = 4

(d) *max_attacks* = 5 (e) *max_attacks* = 6 (f) *max_attacks* comparison

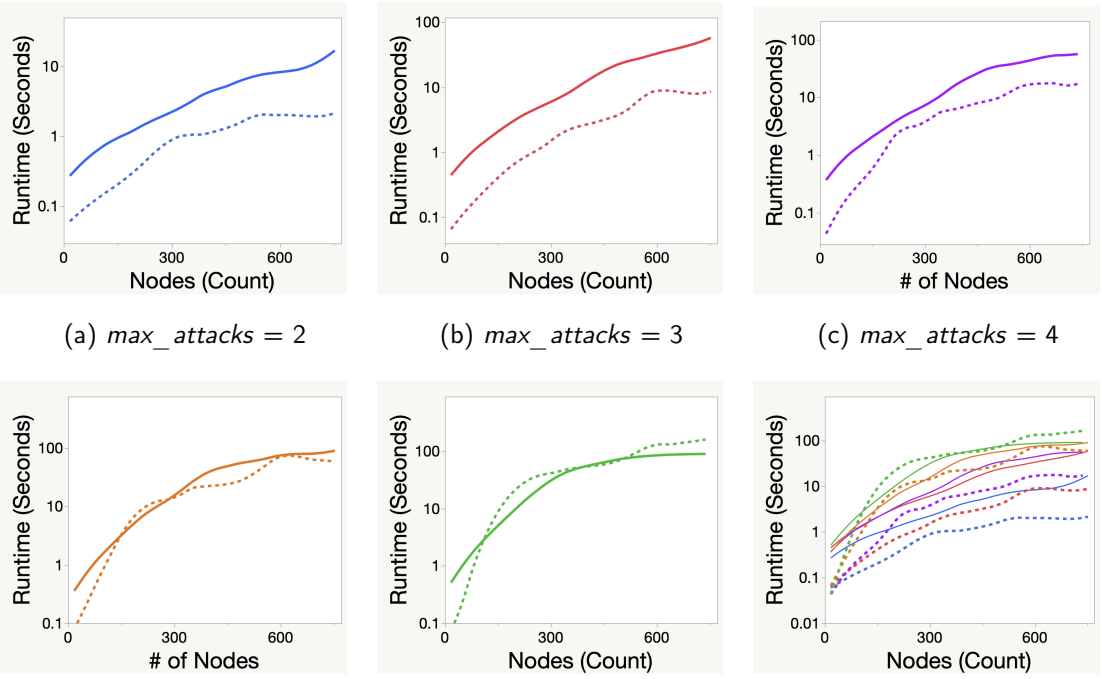
Figure 3.5: Runtime comparison: DILP shortest path on a layered network.

3.3.1 BD vs. DILP

Figures 3.4 and 3.5 and 3.6 show the maximum flow interdiction problem on a grid network at various network sizes and *max_attacks* values.

Our observations indicate that DILP and BD runtimes increase as the network grows and as *max_attacks* grows (see Figures 3.4 and 3.5 . At equivalent *max_attacks* value, the variability of the DILP is higher than BD.

Figures 3.6 also indicates that as *max_attacks* increases, the average runtime for DILP increases more than the average runtime of BD. DILP generally has lower runtimes across tested network sizes at *max_attacks* ≤ 4 (Figure 3.6(a)-(c)), however, for *max_attacks* > 4 (Figure 3.6(d),(e)), the two algorithms perform comparably.



(a) *max_attacks* = 2 (b) *max_attacks* = 3 (c) *max_attacks* = 4

(d) *max_attacks* = 5 (e) *max_attacks* = 6 (f) *max_attacks* comparison

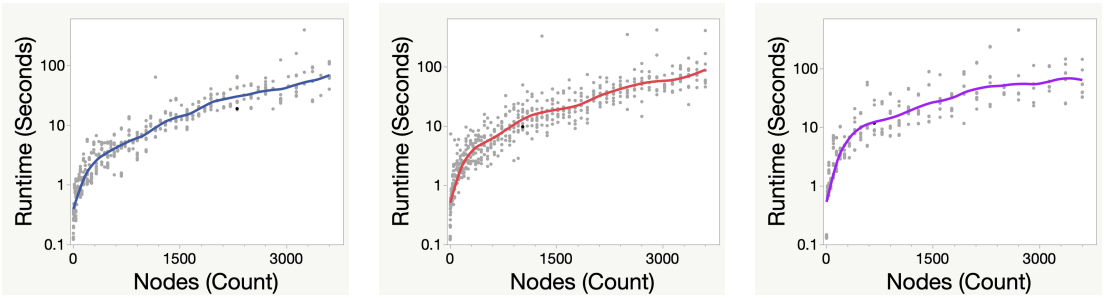
Figure 3.6: Runtime comparison: BD (—) and DILP (- - -) for the shortest path interdiction problem on a layered network.

3.4 Maximum Flow – Grid Network

Figures 3.7, 3.8, and 3.9 show runtime comparisons for the maximum flow problem applied to a grid style network at various network sizes and *max_attacks* values. The increase in nodes $|N|$ directly translates to an increase in network size. Network size includes both the manually added arcs and randomly generated diagonal arcs.

3.4.1 BD

Figure 3.7 shows BD runtimes when applied to the maximum flow interdiction problem on a grid network at various network sizes and *max_attacks* values. Our observations indicate a positive relationship between network size and BD runtime. A comparison of each trend line across tested *max_attacks* levels indicates that generally, as the *max_attacks* increases, so does the average runtime.



(a) *max_attacks* = 2 (b) *max_attacks* = 3 (c) *max_attacks* = 4

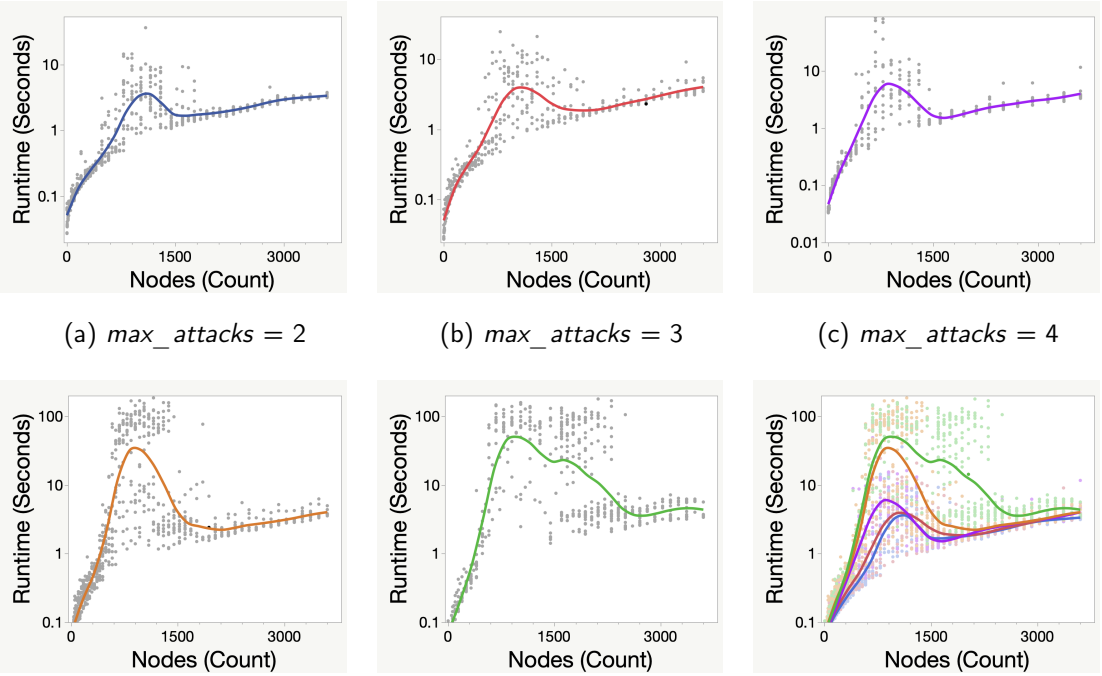
(d) *max_attacks* = 5 (e) *max_attacks* = 6 (f) *max_attacks* comparison

Figure 3.7: Runtime comparison: BD maximum flow on a grid network.

3.4.2 DILP

Figure 3.8 shows DILP runtimes appled to the maximum flow problem on a grid network at various network sizes and *max_attacks* values.

Our observations indicate that, on smaller networks, the DILP runtime increases in a straightforward manner. However, for all *max_attacks* values considered, the DILP algorithm exhibits large variability in runtime for networks containing approximately 900-1,500 nodes (the exact range of this variability depends on the *max_attacks* value). Beyond that point, the variability subsides and the runtimes again exhibit a relatively smooth increase with network size.



(a) *max_attacks* = 2 (b) *max_attacks* = 3 (c) *max_attacks* = 4

(d) *max_attacks* = 5 (e) *max_attacks* = 6 (f) *max_attacks* comparison

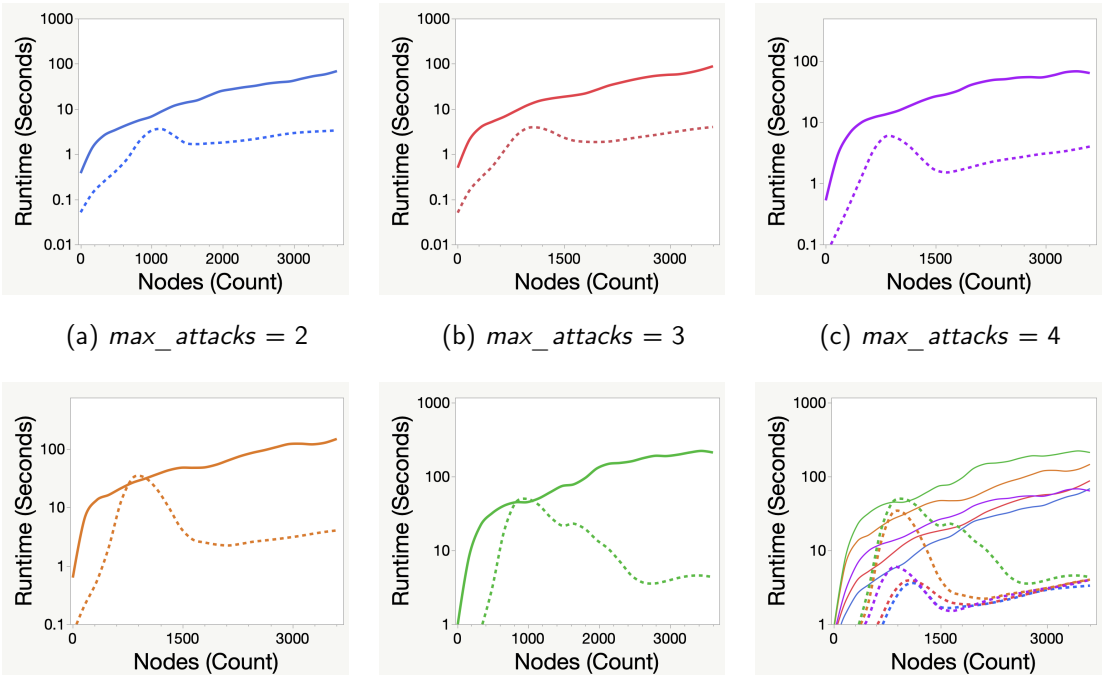
Figure 3.8: Runtime comparison: DILP maximum flow on a grid network.

3.4.3 BD vs. DILP

Figure 3.9 shows a comparison of BD and DILP runtimes when applied to the mmaximum flow interdiction problem on a grid network at various network sizes and *max_attacks* values.

Our observations indicate that the DILP algorithm returns runtimes lower than the BD algorithm in most circumstances.

Both BD and DILP show consistent increases in runtime for relatively small networks, after which the DILP shows a "bump" in runtime due to a subset of instances with very high runtimes. The BD runtime, however, continues to increase smoothly. However, despite the variability in runtime for some network sizes, DILP outperforms BD on average for all settings considered.



(a) *max_attacks* = 2 (b) *max_attacks* = 3 (c) *max_attacks* = 4

(d) *max_attacks* = 5 (e) *max_attacks* = 6 (f) *max_attacks* comparison

Figure 3.9: BD (—) and DILP (- - -) for the maximum flow interdiction problem on a grid network.

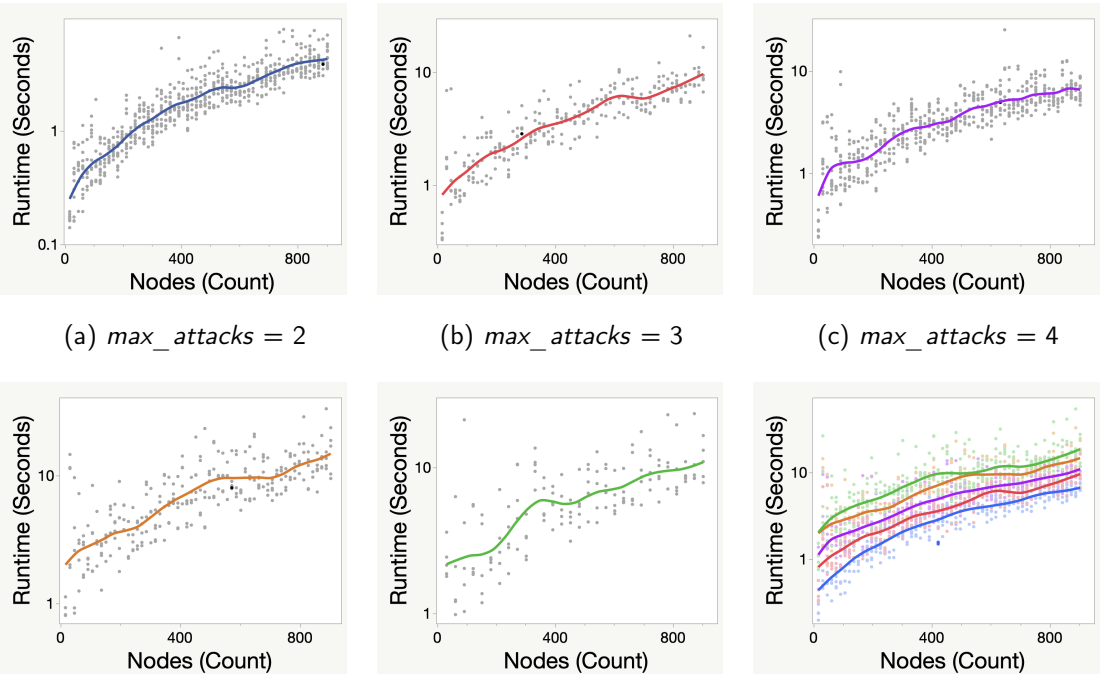
3.5 Maximum Flow – Layered Network

Figures 3.10, 3.11, and 3.12 show runtime comparisons for the maximum flow network problem applied to a layered style network at various network sizes and *max_attacks* values. The size of each layer was fixed at 15 nodes, so the increase in nodes corresponds to an increase in the number of layers present.

3.5.1 BD

Figure 3.10 shows BD applied to the maximum flow interdiction problem on a layered network at various network sizes and *max_attacks* values.

Our observations indicate that there is a positive relationship between network size and BD runtime. A comparison of each trend line across tested *max_attacks* levels indicates that generally, as the *max_attacks* increases, so does the BD runtime (see Figure 3.10(f)).



(a) *max_attacks* = 2 (b) *max_attacks* = 3 (c) *max_attacks* = 4

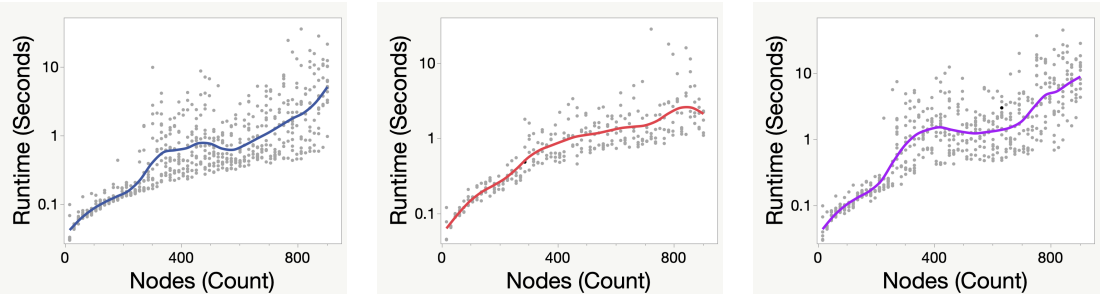
(d) *max_attacks* = 5 (e) *max_attacks* = 6 (f) *max_attacks* comparison

Figure 3.10: Runtime comparison: BD maximum flow on a layered network.

3.5.2 DILP

Figure 3.11 shows DILP applied to the maximum flow interdiction problem on a layered network at various network sizes and *max_attacks* values.

Our observations indicate that as the size of the network grows, the DILP runtime generally increases. However, as with the maximum flow interdiction problem on a grid network, we again observe increased variability in DILP runtime for particular network sizes. This leads to non-monotonic behavior of the corresponding trend lines and is seen across all tested *max_attacks* levels.



(a) *max_attacks* = 2 (b) *max_attacks* = 3 (c) *max_attacks* = 4

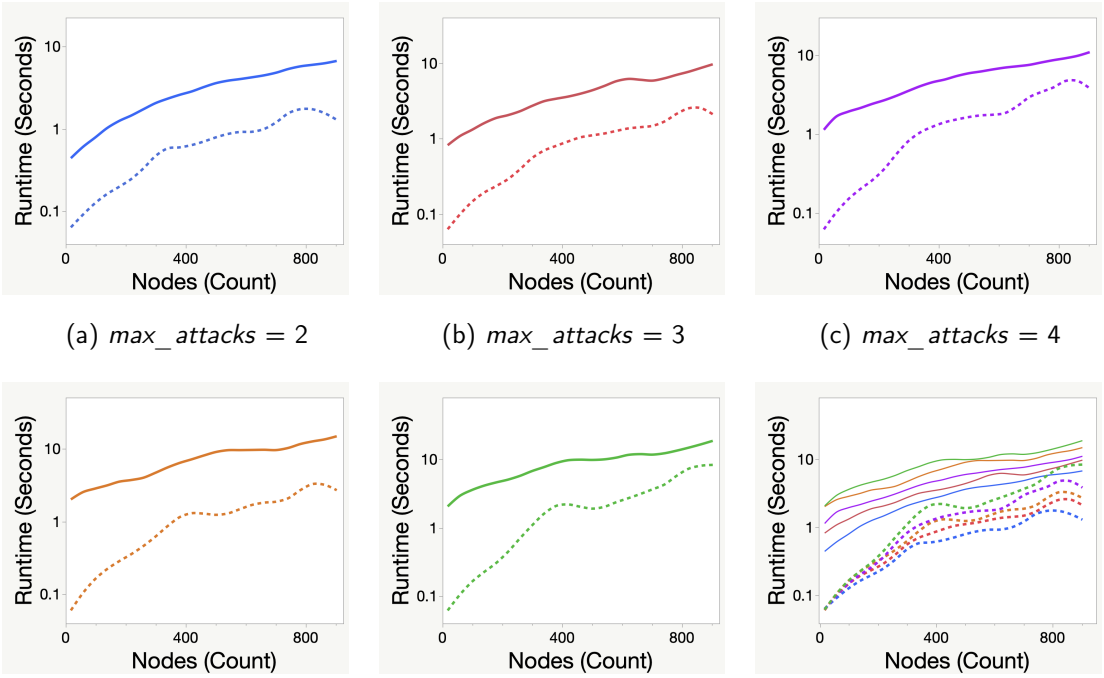
(d) *max_attacks* = 5 (e) *max_attacks* = 6 (f) *max_attacks* comparison

Figure 3.11: Runtime comparison: DILP maximum flow on a layered network.

3.5.3 BD vs. DILP

Figure 3.12 shows a comparison of BD and DILP runtimes when applied to the maximum flow interdiction problem on a layered network at various network sizes and *max_attacks* values.

Our observations indicate that, in general, DILP exhibits lower runtimes than BD for the maximum flow interdiction problem on a grid network (see Figure 3.12). However, as the network size increases, the relative difference in average runtimes decreases significantly. This indicate that convergence between the two algorithms' runtimes may occur at larger network sizes, and BD may outperform DILP for very large networks. As *max_attacks* increases, the two algorithms' runtimes appear to converge at a faster rate.



(a) *max_attacks* = 2 (b) *max_attacks* = 3 (c) *max_attacks* = 4

(d) *max_attacks* = 5 (e) *max_attacks* = 6 (f) *max_attacks* comparison

Figure 3.12: BD (—) and DILP (- - -) for the maximum flow interdiction problem on a layered network.

THIS PAGE INTENTIONALLY LEFT BLANK

CHAPTER 4:
Conclusion

4.1 Summary

This thesis analyzes runtimes for two common network flow interdiction algorithms, BD and DILP, implemented on two practically meaningful network structures. The BD and DILP algorithms each have niches where they perform better. BD generally performs better than DILP when finding the shortest path on a grid network, particularly at higher *max_attack* values. DILP consistently returns lower runtime values for finding the maximum flow on both the grid and layered style networks. However, DILP can exhibit higher variability in rumtime in some settings.

We find that *max_attacks*, network size, network type, and interdiction problem all impacted which algorithm had the lower average runtime.

When finding the shortest path at *max_attacks* ≤ 4 values, our results indicate BD performs better on the grid network, while DILP performs better on the layered network. For *max_attacks* ≥ 4, BD performed better when finding the shortest path on the grid network and both algorithms perform comparably when finding the shortest path on the layered network.

4.2 Recommendations

In general, BD should be used when finding the shortest path on a grid network for *max_attacks* ≥ 4 or when the *max_attacks* value is unknown. DILP should generally be used when finding the shortest path on a layered network for *max_attacks* ≤ 4, and under most circumstances when identifying the maximum flow on either a grid network or a layered network.

4.3 Future Work

The networks considered in this thesis were relatively consistent in density across samples. We utilized straightforward implementations of the two interdiction algorithms that were not specialized to account for any aspects of the network structure. Future analysis might include different meaningful network structures, more nuanced and novel network flow problems, or different interdiction algorithms.

Adding complexity to the network may aid in characterization of algorithms' runtimes. Adding levels of network complexity may include intersecting multiple networks (i.e., differing modes of transportation), multiple source and terminal nodes, and flowing different types of supplies across the network.

List of References

Bartolacci M, Dimitrov S (2017) Promoting resiliency in emergency communication networks: A network interdiction stylized initial case study model of a Miami-Dade County network. *International Journal of Information Systems for Crisis Response and Management (IJISCRAM)* 9(1):10.

Brown G, Carlyle M, Salmeron J, Wood K (2006) Defending critical infrastructure. *Interfaces* 36(6):530–544.

Cormican K, Morton D, Wood K (1998) Stochastic network interdiction. *Operations Research* 46(2):184–197.

Fischetti M, Salvagnin D, Arrigo Z (2010) A note on the selection of Benders cuts. *Mathematical Programming: Heidelberg* 124:175–182.

Ford L, Fulkerson D (1962) *Flows in networks* (Princeton, NJ, Princeton University Press).

Garcia M (2001) *Design and Evaluation of Physical Protection Systems* (Butterworth-Heinemann, Wodburn, MA.).

Golden B, Ball M (1978) Shortest paths with Euclidean distances: An explanatory model. *Networks* 8:297–314.

Heisler Y (2015) The amazing algorithm behind Google maps and Apple maps. https://bgr.com/general/google-apple-maps-engine-algorithm/.

Helmbold R (1971) *A Countercapacity Network Interdiction Model* (RAND Corporation, Santa Monica, CA).

Holmberg K (1990) On the convergence of cross decomposition. *Mathematical Programming* 47:269–296.

INRIX (2019) Inrix: Congestion costs each American 97 hours, $1,348 a year. https://inrix.com/press-releases/scorecard-2018-us/.

Ishimatsu T, de Weck O, Hoffman J, Ohkami Y, Shishko R (2016) Generalized multicommodity network flow model for the Earth–Moon–Mars logistics system. *Journal of Spacecraft and Rockets* 53:25–38.

Lubore S, Ratliff H, Sicilia G (1971) Determining the most vital link in a flow network. *Naval Research Logistics Quarterly* 18(4):497–502.

Lubore S, Ratliff H, Sicilia G (1975) Determining the n most vital link in a flow network. *Management Science* 21(5):531–539.

Preston C (1972) *Interdiction of a transportation network*. Master's thesis, Naval Postgraduate School, Monterey, CA.

Ross J (2014) *Defending critical infrastructure against deliberate threats and non-deliberate threats*. Master's thesis, Naval Postgraduate School, Monterey, CA.

Steinrauf R (1991) *A network interdiction model*. Master's thesis, Naval Postgraduate School, Monterey, Ca.

Svitek P (2021) Texas puts final estimate of winter storm death toll at 246. https://www.texastribune.org/2022/01/02/texas-winter-storm-final-death-toll-246/.

Van Roy T (1983) Cross decomposition for mixed integer programming. *Mathematical Programming* 25:46–63.

Wollmer R (1963) *Some Methods for Determining the Most Vital Link in a Railroad Network* (RAND Corporation, Santa Monica, CA).

Wollmer R (1964) *Removing Arcs From a Network* (Operations Research Center, University of California, Berkeley).

Wollmer R (1970) Determining the n most vital link in a flow network. *Operations Research* 18(3):497–515.

Wood K (1993) Deterministic network interdiction. *Mathematical Computation Modelling* 17:1–18.

Wood K (2002) Shortest-path network interdiction. *Networks* 40(2):97–111.

Wood S (2017) *Generalized Additive Models: An Introduction with R, Second Edition* (CRC Press, Boca Raton, FL).

Initial Distribution List

1. Defense Technical Information Center
   Ft. Belvoir, Virginia

2. Dudley Knox Library
   Naval Postgraduate School
   Monterey, California