MULTILEVEL DISCRETE FORMULATIONS AND ALGORITHMS WITH APPLICATIONS TO NEW PRODUCTION INTRODUCTION GAMES AND NETWORK INTERDICTION PROBLEMS

By

MEHDI HEMMATI

A DISSERTATION PRESENTED TO THE GRADUATE SCHOOL
OF THE UNIVERSITY OF FLORIDA IN PARTIAL FULFILLMENT
OF THE REQUIREMENTS FOR THE DEGREE OF
DOCTOR OF PHILOSOPHY

UNIVERSITY OF FLORIDA

2013

I dedicate this work to

Behjat Ghafouri, the kindest mother (RIP),

Houshang Hemmati, the most wonderful father,

Firouzeh Arbab, the most supportive step-mother, and

Sahar and Sina Hemmati, the best-ever siblings.

ACKNOWLEDGMENTS

I would like to express my sincerest gratitude to my wonderful advisor, Dr. J. Cole Smith, for his invaluable support during my studies at the University of Florida. Words cannot explain my appreciation for his patience, kindness, and enthusiasm in guiding me through my doctoral research as well as his extraordinary help in matters far beyond my research. His encouragement has truly helped me to survive some tough days in my life not solely as a PhD student, but rather as a person. I never felt being framed in an ordinary advisor-student relation, but rather it was an amazing friendship between two persons, one wise and mature and the other one inexperienced.

I also would like to thank Dr. Joseph P. Geunes, Dr. Jean-Philippe P. Richard, and Dr. My T. Thai for serving on my supervisory committee and providing insightful viewpoints and suggestions. In particular, I am thankful for having the chance of taking operations research related courses with Dr. Richard, who is one of the most amazing teachers I could have in my life. I owe many of my related knowledge to him.

Studying here at the University of Florida gave me the opportunity to know some of my most wonderful friends and enjoy my times as a PhD student. Special thanks to my wonderful brothers, Ehsan Salari and Behnam Behdani, for helping me from the very first day of being in Gainesville. I owe them many days of unbelievable support. I also would like to express my appreciation for having the chance to enjoy my times with Cinthia C. Perez, Clay Koshnick, Micheal C. Prince, Andrew N. Romich, Johanna Amaya, Kelly M. Sullivan, and Jorge Sefair with whom I have had many unforgettable memories. I truly appreciate their help and support. I also would like to thank my other friends, Siqian Shen, Aye-nur Arslan, Deon Burchet, Shantih Spanton, Bita Tadayon, Chrysafis Vogiatzis, Jose Walteros, Zehra Melis Teksan, Ruiwei Jiang, Dmytro Korenkevych, Alexey Sorokin, and Reza Skandari for being wonderful colleagues.

Finally, I would like to thank my family. Their unconditional love and endless support have made me who I am. I would be certainly lost in my life journey without them.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

Abstract of Dissertation Presented to the Graduate School
of the University of Florida in Partial Fulfillment of the
Requirements for the Degree of Doctor of Philosophy

MULTILEVEL DISCRETE FORMULATIONS AND ALGORITHMS WITH APPLICATIONS TO NEW PRODUCTION INTRODUCTION GAMES AND NETWORK INTERDICTION PROBLEMS

By

Mehdi Hemmati

August 2013

Chair: J. Cole Smith
Major: Industrial and Systems Engineering

We study multilevel optimization and network interdiction theory, and apply this theory across several applications. The common theme of these problems involves competitive settings in which the beneficiary of a system seeks robust design, protection, or fortification actions to compete against external factors (e.g., uncertainty or an intelligent player) that may affect the system.

First, we consider an extension to the optimal stopping problem in which offered items are also associated with profits from the sellers viewpoint. We seek an ordering that induces the customer to purchase the item that maximizes the sellers profit. By incorporating uncertainty in the problem parameters, we study two optimization philosophies, the max-min profit and the maximum expected profit, and investigate the computational tractability of the resulting optimization models.

Next, we consider a Stackelberg game that arises in new product introduction in which two firms, a leader and a follower, compete with the aim of maximizing their profit by introducing their set of products. Knowing that the success of new products offered by the leader also depends on the followers response, we solve a mixed-integer bilevel program (MIBMP) to find the set of products to be introduced by the leader with the aim of maximizing its profit.

Third, we study a competition scenario over a network in which an adversarial player aims to spread its influence on the nodes over a number of time stages, while the other player aims to protect the network against the spread of the adversarys influence. In this game, the network incurs damage for each influenced node, and the defenders aim is to minimize the maximum damage incurred by the network. We suggest an MIBLP for this interdiction problem, and we propose a cutting-plane algorithm with several valid inequalities.

Our approach to solve these problems employs reformulations and decomposition techniques to devise formulations that are amenable to specially-tailored cutting-plane methods. Moreover, we investigate various separation problems to strengthen these cutting planes. We also conduct computational studies by using CPLEX as the mixed-integer solver and, when applicable, we compare the efficiency of our methods with existing approaches in the literature.

CHAPTER 1
INTRODUCTION

Mathematical programs have been traditionally introduced for applications that involve a single decision maker, who aims to choose a best decision among (possibly a significantly large) number of feasible solutions. Although many classical optimization problems with a single decision maker are considered theoretically intractable, significantly large instances of these optimization problems can now be solved due to the advent of improved computational power and state-of-the-art commercial software packages, which employ vital advances in the related theory. Accordingly, problems having hundreds of thousands variables and constraints can now be solved within reasonable computational limits.

The world of optimization problems, however, also encompasses applications that involve two or more decision makers. For example, applications that address competition between active business agents that aim to maximize their market share involve two decision makers. Here, each agent's decisions must be optimal with respect to not only its own constraints, but also with respect to the other agent's decisions (that may intentionally or unintentionally restrict the first agent's available decisions). These problems are called multilevel optimization problems, and considered to be highly intractable even on small instances. The situation is aggravated in the presence of discrete variables. In this case, multilevel optimization problems often cannot be directly solved using available commercial software packages unless the problem is somehow converted via reformulation into a single-level problem.

Closely related to multilevel problems are optimization problems having uncertain parameters, particularly when a conservative decision maker seeks optimal policies to mitigate the effects of the worst possible outcomes. One framework to study these problems is to perceive an external agent that aims to induce a realization of probabilistic parameters, so that the decision maker faces the worst possible outcomes

depending on his or her decisions. We refer to these problems as interdiction problems. The interdiction modeling paradigm is a powerful mechanism to study conservative decision making in several applications such as homeland security and disaster planning. Stochastic programming and robust optimization are two other widely-known modeling paradigms that allow incorporation of uncertainty in optimization problems. While the former aims, for example, to mitigate the undesired effects of uncertainty over a long-term horizon, the latter seeks policies that are optimal with respect to worst-case scenarios. In fact, robust optimization and interdiction theory employ a very similar point of view in modeling conservative decision making scenarios.

In this dissertation, we primarily focus on multilevel optimization, interdiction theory, and related applications. In particular, we study applications arising in finite optimal stopping under uncertainty, network interdiction, product introduction, and competitive facility location. We propose various bilevel programs having discrete variables in both levels, and we study theoretical characteristics of mathematical formulations as well as efficient solution techniques. Note that Chapters 2–4 are each individually self-contained. Accordingly, the notation required for each chapter are introduced at the beginning of the chapter. This enables the reader to study chapters independently.

In Chapter 2, we study an extension to the finite optimal stopping problem. In this problem, a customer aiming to buy one item receives a series of product offers from a seller one by one. The customer is aware of the (finite) number of product offers and the minimum and maximum possible values of each item, and must purchase exactly one item. When an item is presented to the customer, (s)he observes its value, and determines whether to purchase the item or to permanently dismiss the item. The customer's objective is to maximize the value of the purchased item. In our study, we consider the problem from the viewpoint of the seller, who wishes to maximize profit associated with the sold item. Hence, the seller seeks an optimal sequence of items to sell, given that the customer acts according to some near-optimal decision-making rules.

Our study takes the perspective that the customer may not act optimally due to imperfect decision-making strategies and/or to the seller's uncertainty in the items' values to the customer. We investigate different optimization philosophies for the seller by considering max-min and max-expectation objectives when customer behavior is not completely predictable, and discuss the problem tractability in these cases.

In Chapter 3, we examine a mixed-integer bilevel programming (MIBLP) problem for a competitive set covering problem. The class of problems we consider is applicable to several fields, including non-cooperative product introduction and facility location games. In the prioritized set covering problem, there exists a set of items and clauses. Items may correspond to potential facility locations or products that can be introduced to a market. The clauses may be associated with customers or market segments, each of whom prioritizes the set-cover items according to their interest in these items. We consider a two-player Stackelberg game, in which the leader selects a set of items, and then the follower selects another set of items with knowledge of the leader's action. Every selected item incurs a cost to the players. Each clause is satisfied by the selected item having the highest priority, resulting in a reward for the player that introduced the highest-priority selected item. In this problem, each player aims to optimize its own objective, in contrast to a prior product introduction study in which the follower attempts to minimize the leader's profit. We develop an MIBLP model for this problem in which binary decision variables appear in both stages of the model. As the main contribution of this chapter, we then propose several variations of an exact cutting-plane algorithm to solve this problem, and examine the efficacy of the methods on randomly generated test instances.

In Chapter 4, we consider a bilevel defender-attacker game that takes place on a network, in which the attacker seeks to take control over (or "influence") as many nodes as possible. The defender acts first in this game by protecting a subset of nodes that cannot be influenced by the attacker. With full knowledge of the defender's

action, the attacker can then influence an initial subset of unprotected nodes. The influence then spreads over a finite number of time stages, where an uninfluenced node becomes influenced at time t if a threshold number of its neighbors are influenced at time $t-1$. The attacker's objective is to maximize the weighted number of nodes that are influenced over the time horizon, where the weights depend both on the node and on the time at which that is influenced. This defender-attacker game is especially difficult to optimize, because the attacker's problem itself is NP-hard, which precludes a standard inner-dualization approach that is common in many interdiction studies. We provide three models for solving the attacker's problem, and develop a tailored cutting-plane algorithm for solving the defender's problem. We then demonstrate the computational efficacy of our proposed algorithms on a set of randomly generated instances.

CHAPTER 2
FINITE OPTIMAL STOPPING PROBLEMS: THE SELLER'S PERSPECTIVE

2.1 Introduction and Literature Study

We begin this chapter by describing the following stopping problem, which is a classical problem that has received substantial attention across several disciplines. A customer must purchase one item out of a set of items that are presented one at a time to the customer. When an item is presented to a customer, the customer evaluates its value, and must decide whether to purchase the item (thus ending the game) or permanently discard (or "reject") the item. The customer's objective is to maximize the value of the purchased item. (The reward is equal to the value of the purchased item, and is independent of all other items' values.) The customer is aware at the beginning of the game of the number of items (denoted by n) that will be presented and the probability distribution used to generate the items' values. Observe that if one item remains, the customer must purchase it. It is thus straightforward to see that this model captures the case in which the customer does *not* need to buy an item, which we would allow by simply letting the final item have a value equal to the customer's value of purchasing nothing at all.

This game is a special case of the broad class of *optimal stopping problems* in which the customer must determine when to stop the game (e.g., by purchasing an item). See [20] for an early summary of optimal stopping problem analysis, and [32] for a comprehensive modern study of this class of problems. In fact, the original "secretary problem" (see, e.g., [31, 33, 34, 59]) is given as above, but where the objective is to pick the most-valuable item from among the set of all items. There is no reward for picking any other item than the best one. There are numerous versions of these games, including variations in which n is unknown or infinite, in which rewards are given for solutions other than the most-valuable one, and in which an attempt to purchase the item may fail [4, 37, 57, 69].

The stopping problem considered in this chapter satisfies the so-called memoryless property, in the sense that prior actions that occurred in this game do not affect the remainder of the game. The customer only needs to know how many items remain and the value of the next item (in addition to the boundary conditions of the game) to make the next decision; prior information regarding the values of previous (rejected) items is irrelevant. This memoryless property enables us to employ dynamic programming techniques (see, e.g., [8]) to solve the above stopping problem. (We provide the technical details for this algorithm in Section 2.2.)

In this chapter, we introduce a new problem from the seller's perspective. In this problem, each item is also associated with a *profit* (independent from the item's *value* to the customer) that the seller makes if the customer purchases the item. The seller must determine a sequence of the items to present to the customer, so that the customer (acting rationally, i.e., optimally in his/her own best interests) would choose an item that results in a maximum possible profit to the seller. This problem is nontrivial, because placing the most-profitable items too early in the sequence may result in the customer rejecting those items, in hopes of finding an item with more value to the customer later in the sequence. Placing the most-profitable items late in the sequence incurs the risk of having the customer terminate search before encountering these items.

Moreover, this problem is further compounded the fact that human decision-makers tend *not* to optimally solve stopping problems in laboratory settings. We refer the reader to an excellent introductory treatment of this material in [6], and to recent results appearing in [7, 43, 61]. A common theme in this line of literature is that decision-makers tend to terminate their search too soon. It is potentially very risky for the seller to assume that a customer will act rationally, in the sense that the customer follows an exact optimization algorithm in selecting an item.

We present an example to illustrate the situation. Suppose that $n = 6$, and that the customer believes that the items' values are uniformly distributed in the interval $[0, 100]$.

Items $1, \ldots, 6$ are arranged in nonincreasing order of profit to the seller (so that item 1 is most preferred). Consider the situation given in Figure 2-1, which depicts a sequence that the seller has chosen to present to a customer. The seller has evidently gambled that the customer will reject item 5, which has a value of 70 to the customer. Ideally, the customer would then purchase item 1, resulting in the most profit to the seller.

Indeed, an optimal customer will reject item 5 with six items remaining, and purchase item 1 with five items remaining. (In Section 2.2, we will illustrate an optimal decision-making framework for the customer. In particular, it turns out that an optimally behaving customer would buy the sixth-to-last item if its value exceeds 77.5, and would buy the fifth-to-last item if its value exceeds 74.2.) However, the seller risks having a conservative decision-maker (rather than an optimal customer) that stops the process too early and purchases item 5, or stops the process too late and purchases item 6.

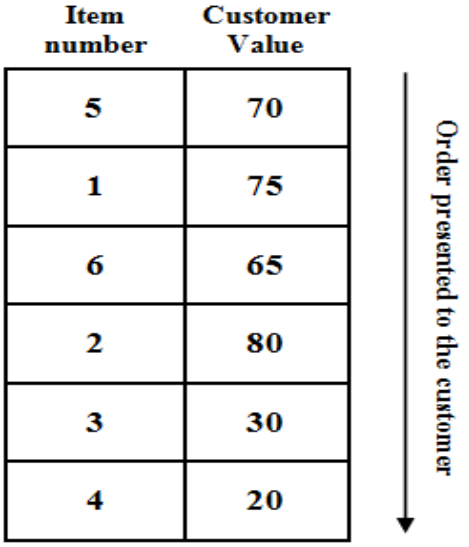

Figure 2-1. Sequence of items in an optimal stopping problem.

The challenge in this chapter is to analyze the seller's problem from three different perspectives. We begin in Section 2.2 by assuming that the customer acts in an optimal manner, breaking ties in a manner that is disadvantageous to the seller. That is, when the customer's decision is optimal either to reject or purchase an item, the customer

takes the opposite action desired by the seller. (This pessimistic assumption can easily be modified.)

We then accommodate the stochastic nature of human decision-makers by accounting for randomness in the customer's optimal decision-making policies, and in the customer's perception of the items' values. For instance, we can model a decision-maker who tends to stop too soon by adjusting the true item values to be higher than they actually are. However, when dealing with uncertainty, we must specify an objective that reflects the seller's optimization philosophy. A risk-averse seller may attempt to maximize the minimum profit that can be made, given a boundedly rational customer. We discuss this problem, along with a simple model for bounding customer rationality, in Section 2.3.

Although a seller who is playing this game once may prefer an outcome with limited risk of selling a low-profit item, a seller who plays this game repeatedly would more likely prefer to maximize *expected* profit instead. Hence, in Section 2.4, we present a problem in which the seller maximizes expected profit given a discrete probability distribution function of the customer's problem-solving parameters. We demonstrate that even restricted versions of the expected-value problem are NP-hard, and are thus quite difficult to solve in the worst case.

2.2 Seller's Problem with an Optimal Customer

We begin by introducing notation for this problem and presenting the optimal dynamic programming strategy for the customer. For each item $i = 1, \ldots, n$, the item's value to the customer is v_i, and the item's profit to the seller is b_i. For the sake of simplicity, we examine the case in which the customer assumes that item values are generated from the uniform distribution on the interval [0,100]. The lower and upper bounds given here are arbitrary, and the following discussion easily accommodates any generic (finite) bounds. Also, the logic behind our procedures does not change if the

values are assumed to be nonuniformly generated, so long as conditional expectations are finite.

As stated in Section 2.1, we assume that the customer decisions are not affected by the values of the previously seen items, but only by the number of remaining items. More precisely, the customer's dynamic programming algorithm employs backward recursion, starting from the situation in which only one item remains. In this case, the customer must purchase the item, and the customer's expected value will be 50. Now, if there are two items remaining in the set, the customer will purchase the item if its expected value is at least 50, and will reject the item otherwise. When two items remain, there is a 50% chance that the customer rejects the item, because its value does not exceed 50 (leaving the customer with an expected item value of 50 from the last item), and a 50% chance that the customer accepts the item because its value belongs to the interval [50,100] (yielding an expected value of 75 from the second item). The customer's overall expected value with two items left is thus 62.5.

In general, when examining the ith item in the sequence, we define t_i to be the customer's expected value from purchasing an item. A rational (or *optimal*) customer applies the following recursive formula to compute these values:

$$t_n = 50, \tag{2–1a}$$

$$t_i = \Pr\left(V > t_{i+1}\right)\left(\frac{t_{i+1}+100}{2}\right) + \Pr(V \le t_{i+1})\, t_{i+1} \forall \qquad i = 1, \ldots, n-1, \tag{2–1b}$$

where V is the uniform random variable reflecting the value of any item; hence, $\Pr(V \le k)$ is 1 if $k \ge 100$, 0 if $k \le 0$, and $k/100$ otherwise, with $\Pr(V > k) = 1 - \Pr(V \le k)$. (Note that this process can be adapted for nonuniform distributions, and for distributions that depend on how many items remain.) Hereafter we refer to t-values as *thresholds*, because they reflect how an optimal customer makes decisions: If the ith item's value offered to the customer exceeds t_{i+1}, then the customer selects it. For notational

convenience (allowing us to handle the case corresponding to the last item), we define $t_{n+1} = -\infty$.

Without loss of generality, we assume $b_1 \geq \cdots \geq b_n$. We discuss the seller's problem in terms of assigning items to slots in the sequence presented to the customer. The seller's goal will be to induce the customer to buy item 1 if possible, and failing that, to buy item 2, and so on. Conceptually, the seller will place item 1 in the earliest slot i such that $v_1 > t_{i+1}$, and item 1 is purchased if the customer reaches slot i in the search. Accordingly, we define

$$p_i = \min\{p \in \{1, \ldots, n\} : v_i > t_{p+1}\} \quad \forall i = 1, \ldots, n. \tag{2–2}$$

For item $i = 1, \ldots, n$, p_i represents the earliest possible slot in any sequence in which the customer would choose item i, assuming that the search is still active. Of course, if $p_1 = 1$, the problem is trivial: Place item 1 in the first slot, with the remaining slots arbitrarily determined, and the customer selects item 1. Otherwise, the seller seeks to place less-profitable items early in the sequence whose values are small enough that the customer rejects them, until reaching item 1. Define:

$$N_p = \{i \in \{1, \ldots, n\} : v_i < t_{p+1}\} \quad \forall p = 1, \ldots, n. \tag{2–3}$$

The set N_p contains the items that would not be chosen by the customer if they are positioned in slot p. The following proposition is useful in devising our solution method:

Proposition 2.1. *Suppose that no solution exists in which an item in* $\{1, \ldots, i-1\}$ *can be sold to the customer, for some* $1 \leq i < n$*, and define* $\Gamma = \min\{p_i - 1, i - 1\}$*. Then item* i *can be sold to the customer if and only if a sequence exists in which item* i *is placed at slot* p_i*, and items* $j = 1, \ldots, \Gamma$ *are placed in any arbitrary order in slots* $p_i - \Gamma, \ldots, p_i - 1$*. Also, if* $p_i > i$*, then an item in* N_p *is placed at slot* p*, for all* $p = 1, \ldots, p_i - i$*, with no item appearing twice in slots* $1, \ldots, p_i$*.*

Proof. First, suppose that the conditions of this proposition hold true. The customer will purchase item i positioned at p_i if no item is selected before p_i. By assumption the customer rejects any item $h < i$. The only items $j > i$ placed in a slot $p < p_i$ obey the condition $j \in N_p$, and so the customer will reject these items as well, thus purchasing item i in slot p_i.

Now, suppose that it is possible to sell item i (but not items in $\{1, \dots, i-1\}$). We first establish that a schedule exists in which item i is placed in position p_i. If i is placed before p_i, the customer will not purchase it. Else, suppose that a schedule exists in which item i is placed in slot $k > p_i$. Then by switching the items in slots k and p_i, the customer would still purchase item i in p_i. Given that item i is placed in slot p_i, we next establish that a schedule must exist in which each item $j \in J = \{1, \dots, \Gamma\}$ is placed in slots $p_i - \Gamma, \dots, p_i - 1$ (in any order). Because items in J cannot be purchased by the customer, they can be placed anywhere in the sequence. Suppose that an item $j \in J$ is not one of the $|J|$ items that immediately precedes item i in position p_i, and swap j with an item $k \notin J$ that is currently positioned in some slot $p \in \{p_i - \Gamma, \dots, p_i - 1\}$. If j had followed item i, then the customer would reject j instead of k in slot p. Otherwise, $p \in \{1, \dots, p_i - i\}$, and item k is moved earlier in the sequence. If item k was rejected in its original position, it would also be rejected at an earlier position. The customer still rejects item j (in any position), and because the remaining items are unaffected, would still purchase item i. After executing all such swaps, and rearranging items in J as necessary, we obtain a schedule satisfying the conditions of the proposition. □

Using Proposition 2.1, the seller can employ the following greedy principle. Starting with item 1, examine whether there exists a sequence in which item 1 is placed in p_1 and the earlier positions $p = 1, \dots, p_1 - 1$ contain items belonging to N_p. If so, an optimal solution can be constructed based on this sequence. Otherwise, the seller attempts to sell item 2, and so on, until an item can be sold.

Computationally, our challenge is to solve the problem of placing items in slots $1, \ldots, p_i - i$ (when $p_i > i$) as efficiently as possible. First, it is useful to define a maximum "safe" slot for each item i as:

$$f_i = \begin{cases} 0 & \text{if } v_i \geq t_2 \\ \max\{p \in \{1, \ldots, n\} : v_i < t_{p+1}\} & \text{otherwise.} \end{cases} \qquad \forall i = 1, \ldots, n, \tag{2–4}$$

which indicates that item i cannot be purchased in slots $1, \ldots, f_i$, where $f_i = 0$ means that item i would be purchased in any slot. (Note that $f_i = p_i - 1$ if $v_i \neq t_p$ for all $p = 2, \ldots, n$.) Now, suppose that we sort the f-values into a list F in nondecreasing order of their values, discarding those i for which $f_i = 0$. We now state the following lemma, related to ordering the first items of a sequence.

Lemma 1. *Suppose that no solution exists to a given problem in which any item in the set* $\{1, \ldots, i-1\}$ *can be purchased by the customer. Further suppose that* $p_i > i$*, and that there exists an (optimal) sequence with item* i *placed in slot* p_i*, which induces the customer to purchase item* i*. Then there exists a solution in which items scheduled in positions* $1, \ldots, p_i - i$ *appear in the same order as they appear in* F*.*

Proof. Consider a sequence with i positioned in p_i, and a subset of items in the set $\{i+1, \ldots, n\}$ in positions $1, \ldots, p_i - i$ (such a solution is known to exist by Proposition 2.1). Items in the first $p_i - i$ slots having the same f-value can clearly be rearranged to meet the sequence of items in F. Now, consider items j and k with $f_j > f_k$, but item j (in slot p') ordered before item k (in slot p''). Suppose that we swap these items' slots. Item k in slot p' would still be rejected, because item k was rejected in the later slot of p''. Item j would be rejected in slot p'' because k was rejected in the same slot, and the fact that $f_j > f_k$ indicates that $v_j < v_k$. With a lesser value to the customer, the customer would also reject j in slot p''. □

We can now execute the following algorithm to optimize the seller's problem, which considers only sequences of the form given in Proposition 2.1, whose first $p_i - i$ slots

(if $p_i > i$) are scheduled according to Lemma 1. We formally state this process in Algorithm 1.

Algorithm 1 Seller's Problem with an Optimal Customer

Compute p_i and f_i, $\forall i = 1, \ldots, n$. Obtain F by sorting the items in nondecreasing order of their values, excluding item 1, and breaking ties in decreasing order of item index.
Initialize $j = 1$ and $\texttt{list} = \emptyset$; let $|\texttt{list}|$ denote the number of elements in $\texttt{list}$.
while $j < n$ **do**
  Pick the next item of F, and call it item k.
  if $f_k \geq |\texttt{list}| + 1$ **then**
    Add k to the end of $\texttt{list}$.
    Remove item k from F.
  end if
  Set $j = j + 1$.
end while
Set $j = 1$.
while $p_j > j + |\texttt{list}|$ **do**
  Set $j = j + 1$.
  if item j belongs to $\texttt{list}$ **then**
    Let pos_j be the position of item j in $\texttt{list}$.
    Delete item j from $\texttt{list}$.
    if there exists an item $k \in F$ with $f_k \geq pos_j$ **then**
      Find an item k having the smallest value of f_k in F, subject to $f_k \geq pos_j$.
      Insert item k into position pos_j of $\texttt{list}$, and delete k from F.
    end if
  end if
end while
An optimal sequence schedules items as in Proposition 2.1, with the ordering of the first $\min\{p_j - 1, j - 1\}$ items given by $\texttt{list}$.

The first while-loop in Algorithm 1 establishes $\texttt{list}$, which aids us in maximizing the number of items that are to placed at the front of the sequence. After this first step is complete, the second while-loop terminates when $j + |\texttt{list}|$ is at least as large as p_j: When this happens, scheduling a combination of items on $\texttt{list}$ (in positions $p < p_j$) and items $1, \ldots, j$ results in a sequence in which the customer buys item j. If this condition is not satisfied, then we instead attempt to sell item $j + 1$. But first, if item $j + 1$ had been a part of $\texttt{list}$, which preceded j at the previous iteration, we must remove it from its

tentatively scheduled position. We seek a replacement in the list in Step 18. If such an item exists, the size of `list` remains constant, and otherwise, it shrinks by one.

Each value p_i and f_i can be computed in $O(\log n)$ steps by binary search, for a total of $O(n \log n)$ operations. The sorting operation for computing F takes $O(n \log n)$ time as well. The operations in the first while-loop are $O(n)$ in complexity. In the second while-loop, we must potentially delete an item from `list` (which can be done in constant time), find item k in Step 18 (which requires $O(\log n)$ steps), and perhaps insert an element back into `list` (which can be done in constant time). However, if a position index is kept explicitly at each iteration, the algorithm requires $O(n)$ operations at each update. Instead, in Step 15, we find the position of j in `list` via binary search. (Note that items appear in `list` in the same order that they appeared in F; furthermore, the tie-breaking criteria present in our sorting of F ensures that the sorting operation yields a unique sequence F. These facts permit us to execute binary search on `list`.) Step 15 therefore also takes $O(\log n)$ steps, and so the second while-loop requires $O(n \log n)$ steps. Finally, recovering the sequence at the end of the algorithm is an $O(n)$ operation, and so the overall complexity of this algorithm is $O(n \log n)$.

To illustrate this process, consider the $n = 10$ example whose v-, t-, p-, and f-values are depicted in Table 2.2. Initially, we have

$$F = \{8,\ 6,\ 9,\ 7,\ 5,\ 10,\ 3,\ 2,\ 4\} \text{ and}$$

$$\texttt{list} = \{9,\ 7,\ 10,\ 3,\ 2,\ 4\},$$

where items 8 and 6 do not belong to `list` because $f_8 = f_6 = 0$, and item 5 does not belong to `list` because it would be the third item, and $f_5 = 2 < 3$. Following the creation of `list`, $F = \{8,\ 6,\ 5\}$. We try to find a sequence such that item 1 can be sold, but this is impossible because 1 + |`list`| = 7 and $p_1 = 10$. That is, even if every item in `list` is ordered before 1, the customer would purchase any item (other than 1) in the seventh position.

Table 2-1. Seller's problem example

	1	2	3	4	5	6	7	8	9	10
v_i	30	68	78	60	83	85	83	92	84	81
t_i	86.1	85.0	83.6	82.0	80.0	77.5	74.2	69.5	62.5	50.0
p_i	10	8	5	9	3	1	3	1	2	4
f_i	9	7	4	8	2	0	2	0	1	3

Having ruled out selling item 1, the seller attempts to sell item 2. First, item 2 is removed from `list`, and is not replaced: Item 2 sits in the fifth position of `list`, and no item in F has an f-value of at least 5. Hence, item 2 could only be sold if $p_2 \leq 2 + |\texttt{list}| = 7$ (representing the five-item sequence of `list` being scheduled, followed by item 1, followed by item 2), but $p_2 = 8$ and the item cannot be sold.

Turning our attention to selling item 3, we remove item 3 from `list` (again without replacement), and test $p_3 \leq 3 + |\texttt{list}| = 7$. This time, the relationship holds, and item 3 can indeed be sold to the customer. According to Proposition 2.1 and Lemma 1, one optimal sequence begins with

$$\texttt{sequence: } 9\ 7\ 2\ 1\ 3,$$

with the remaining five items being scheduled arbitrarily.

2.3 Max-Min Problem

The stopping problem we discuss in this chapter essentially encompasses three different parameter sets: profits, thresholds, and customer values. A vital assumption that we made in Section 2.2 is that all parameters are known with certainty, and that the customer will employ an optimal strategy to select an item. However, in a more realistic setting, there may be some degree of uncertainty about the parameters, and (especially in the case of a human decision-maker) about the strategy that the customer uses. The seller should therefore incorporate knowledge of this uncertainty into the sequence. We consider in this section a conservative seller, who seeks a sequence that maximizes

profit in a worst-case scenario. (The case of a seller who wishes to maximize expected profit instead given a particular data distribution is explored in Section 2.4.)

Note that it is not generally possible to specify a single "worst-case scenario" (i.e., an outcome of random data that is most damaging) for the seller. However, given an established sequence of items, it is possible to determine a worst-case set of data that results in the minimum profit for the seller. Hence, in this modeling strategy, we define an uncertainty set $\mathcal{U}$ of all possible combinations of threshold and customer values (with profit values being deterministic). In general, the only assumptions that we make regarding the structure of $\mathcal{U}$ is that it is nonempty, and the threshold values $\bar{t}$ that belong to $\mathcal{U}$ must satisfy $\bar{t}_1 \geq \cdots \geq \bar{t}_n > t'_{n+1} = -\infty$. Note that from a game-theoretic perspective, one can view $\mathcal{U}$ as the set that defines boundedly rational behavior from the customer.

Denote $\ell(x, \mathcal{U})$ as the minimum profit possible given a sequence of items x over all possible data outcomes in $\mathcal{U}$. The problem considered in this section seeks to maximize $\ell(x, \mathcal{U})$ over all permutations of items x, which is why we refer to this problem as a max-min optimization problem. This modeling philosophy is exactly that embodied by the *robust optimization* community; we refer the reader to [9] for a thorough mathematical programming discussion of robust optimization.

Because identifying the worst-case data scenario corresponding to any item sequence x is an optimization problem, it is convenient to envision a third party *adversary* who seeks the worst possible data outcome in $\mathcal{U}$, given the seller's sequence of items. Hence, we now examine this problem as a Stackelberg game in which the seller arranges the items to be sold in some sequence, the adversary manipulates data (within the allowable uncertainty set $\mathcal{U}$), and the customer follows the previously stated optimal stopping strategy for selecting an item, but based on the parameters manipulated by the adversary.

A simple generalization of Algorithm 1 is not sufficient to solve the max-min problem described above. (Indeed, Corollary 1 in Section 2.4 will demonstrate that this problem is NP-hard in general.) However, we illustrate in this section one strategy for solving a max-min problem given a specific class of $\mathcal{U}$-sets. Consider any set in which the threshold values are fixed at their optimal t-values as computed by Equations 2–1a and 2–1b, $\forall i = 1, \ldots, n$, and where the item values v are restricted as follows for some nonnegative integer K:

$$v_i' - \Delta y_i \leq v_i \leq v_i' + \Delta y_i \quad \forall i = 1, \ldots, n \tag{2–5a}$$

$$\sum_{i=1}^{n} y_i \leq K \tag{2–5b}$$

$$y_i \in \{0, 1\} \quad \forall i = 1, \ldots, n. \tag{2–5c}$$

where $\Delta \geq 0$. Here, v_i' acts as a nominal value for each $i = 1, \ldots, n$. Note that Constraints 2–5a state that the true value for item i is somewhere in the interval $[v_i' - \Delta, v_i' + \Delta]$. Constraint 2–5b states that only some K parameters v_i may deviate from their nominal values, and Constraint 2–5c states logical restrictions on the y-variables that control which parameters deviate from their nominal values.

Our approach to solve this problem follows the general strategy employed in Algorithm 1: The seller searches for a sequence that results in some item $i = 1, \ldots, n$ being chosen by the customer, stopping when the highest-profit item can be sold. However, we must now take the adversary's role into account, noting that the customer values may be modified from their nominal values by the adversary to prevent an item from being sold, or induce a (low-profit) item to be sold.

For convenience, we define:

$$J_i^1 = \{1, \ldots, i-1\} \quad \forall i = 1, \ldots, n, \tag{2–6a}$$

$$J_i^2 = \{i+1, \ldots, n\} \quad \forall i = 1, \ldots, n. \tag{2–6b}$$

Also, given a sequence of items, define pos_i as the position of item i in the sequence.

We next provide the following proposition, which establishes the form of optimal sequences given an uncertainty set of the form .

Proposition 2.2. *Let i be the (smallest) index of the item sold to the customer in an optimal sequence given an uncertainty set of the form . An optimal sequence consists of items in sets $A_1, \ldots, A_5$ (some of which may be empty) in the order*

$$A_1 - A_2 - A_3 - A_4 - \text{item } i - A_5 \tag{2–7}$$

such that:

- *A_1 consists of items $j \in J_i^2$ such that $v_j' + \Delta < t_{pos_j+1}$,*
- *A_2 consists of items $j \in J_i^1$, K of which satisfy $v_j' > t_{pos_j+1}$, including the last element of A_2,*
- *A_3 consists of items $j \in J_i^2$ such that $v_j' < t_{pos_j+1}$,*
- *A_4 consists of items $j \in J_i^1$, and*
- *A_5 consists of items $j \in J_i^2$,*

where sets $A_1, \ldots, A_5$ and $\{i\}$ form a partition of the items $\{1, \ldots, n\}$.

Proof. To begin, it is useful to interpret the sets A_i, for $i = 1, \ldots, 5$.

- A_1 consists of items $j \in J_i^2$, which can safely be scheduled so that even if the adversary modifies v_j to take on its largest possible value, the customer will not purchase it.
- A_2 consists of items $j \in J_i^1$ that (by assumption) cannot be sold. Indeed, the adversary has been forced to use all K of its allotted value deviations (i.e., setting $y_i = 1$ for items $i \in A_2$) to prevent the customer from buying these items. In particular, the adversary must have modified the last item's value in A_2 to prevent it from being sold.
- A_3 is similar to A_1, with the exception that these items will not be sold if the adversary cannot modify their values (instead of the stronger condition stated for items in A_1). The placement of these items before item i in the sequence is valid because the adversary used all K modifications to prevent items in A_2 from being sold. Hence, if A_2 is empty, then so is A_3.

- A_4 is similar to A_2, without the caveat that the adversary must take action to prevent their sale.
- A_5 consists of all items in J_i^2 not contained in A_1 or A_3.

Consider any original sequence in which item i can be sold, which does not satisfy 2–7. We show that there also exists a modified sequence satisfying 2–7 such that item i is still sold.

First, consider the case in which the adversary is forced to use K modifications to prevent an item in J_i^1 from being sold in the original sequence, before item i is eventually sold. (For simplicity, we say that the adversary has "attacked" an item in J_i^1 if the adversary sets $v_i = v_i' - \Delta$ to prevent it from being sold.) In the original sequence, let p' be the position of the earliest scheduled item that belongs to J_i^1, and let p'' be the position of the Kth item that is attacked in J_i^1, noting that $p'' < pos_i$. Suppose that an item $j \in J_i^2$ is positioned so that $p' < pos_j < p''$. Note that j satisfies the criterion $v_j' + \Delta < t_{pos_j+1}$, because otherwise, the adversary (not having used all K attacks before position pos_j) could induce the customer to purchase item j. We could therefore swap the position of item j with the item in J_i^1 in slot p', with item i still being sold. Note that j must still satisfy $v_j' + \Delta < t_{p'+1}$ because $t_{p'+1} > t_{pos_j+1}$; the latter inequality also indicates that if the adversary had to attack the item in slot p' in the original sequence, it must still do so when this item is in the later slot pos_j in the modified sequence. After repeating this procedure, all items in A_1 will precede those in A_2.

Furthermore, suppose that the last item in A_2 is not attacked by the adversary. By swapping the order of any item j in A_2 that is attacked with the last item in A_2, we have that item j is still attacked in its later slot. Hence, a sequence exists in which the last item in A_2 is attacked.

Now, say that the last item in A_2 ends at slot q' in the original sequence. If no items in J_i^2 are scheduled after A_2 and before i, then A_3 is empty. Else, suppose that the last item in J_i^2 scheduled before item i is in position q''. If there is no item in J_i^1 positioned

in some slot q such that $q' < q < q''$, then the schedule follows 2–7 as desired. Else, consider such an item j in position q. Suppose that we modify the sequence by swapping the position of items in positions q and q''. The item formerly in position q'' belongs to J_i^2 and cannot be sold at the earlier slot, q. Item j belongs to J_i^1 and cannot be sold by assumption (and is not attacked in the original or modified sequence because it follows the set of items A_2). Hence, item i will still be sold in the modified sequence. Performing all such swaps yields a sequence that satisfies the conditions stated in 2–7 pertaining to items in slots $1, \ldots, pos_i$.

To verify that $A_5 \subset J_i^2$, suppose that an item $j \in J_i^1$ is scheduled after i in the original sequence. A modified sequence would place j in the position of item i, and shift all items placed in slots $pos_i, \ldots, pos_j - 1$ back one spot in the modified sequence. Because item j cannot be selected by the customer, and because item i is appearing later in the modified sequence than in the original sequence (where it was selected), item i will still be sold in the modified sequence. Hence, a sequence exists in which A_5 consists only of items in J_i^2.

Second, we consider the case in which the adversary does not need to use all K modifications in the original sequence to prevent items in J_i^1 from being sold. If the original sequence does not satisfy 2–7, then there exists an item $j \in J_i^1$ scheduled before an item $k \in J_i^2$, which is scheduled before i. By the same argument as above, items j and k can be swapped in a modified sequence, and item i would still be sold in the modified sequence. Also, the same argument showing that A_5 consists only of items belonging to J_i^2 holds as above. Repeating these swaps leads to a sequence that satisfies 2–7, with A_2 and A_3 being empty. □

The following lemmas allow us to further restrict our attention to special sequences that obey 2–7.

Lemma 2. *Consider an optimal sequence of the form 2–7 in which $A_2 \neq \emptyset$, $|A_1| = p$ for some nonnegative integer p, and item i is sold to the customer. Let the items in*

J_i^1 be sorted in nonincreasing order of their nominal values, yielding the sequence $\alpha(1), \ldots, \alpha(|J_i^1|)$. Also, let J_i^2 be sorted in nonincreasing order of their nominal values, yielding the sequence $\beta(1), \ldots, \beta(|J_i^2|)$. Then:

1. *Items in A_1 are ordered in the following greedy fashion: Set $j = 1$ and $q = 1$, and determine if $v'_{\beta(j)} + \Delta < t_{q+1}$. If so, then place $\beta(j)$ in position q and increment q by one. In either case, increment j by one. Stop if $q = p + 1$; else, repeat.*
2. *Items $\alpha(1), \ldots, \alpha(K)$ are scheduled so that the adversary attacks each item to prevent them from being sold. Item $\alpha(1)$ is placed in the earliest position $q > p$ such that $v'_{\alpha(1)} > t_{q+1}$. Then, item $\alpha(j)$ is placed in the earliest position $q > pos_{\alpha(j-1)}$ such that $v'_{\alpha(j)} > t_{q+1}$, for each $j = 2, \ldots, K$. All other positions $p+1, \ldots, pos_{\alpha(K)}$ that have not been assigned an item (if any) are assigned unscheduled items from J_1^i in any arbitrary order.*
3. *Items in A_3 are ordered according to the same greedy algorithm as for A_1, except we start at position $q = |A_1| + |A_2| + 1$, ignore those items that have been scheduled in A_1, and test whether $v'_{\beta(j)} < t_{q+1}$, noting that the adversary cannot adjust the values of items in A_3.*

Proof. Consider any original sequence of the form 2–7 in which $A_2 \neq \emptyset$, $|A_1| = p$, and item i is sold to the customer, but was not generated according to claims 1–3 in the lemma. We again show that a modified sequence exists in which item i is sold that does conform to these three claims.

First, suppose that claim 1 does not hold in the original sequence. Then there exists a minimum index p', with item $j \in J_i^2$ positioned in p', such that a higher-valued item $k \in J_i^2$, $v'_k + \Delta < t_{p'+1}$, is positioned in slot $p'' > p'$. We could generate a modified sequence by swapping the position of items j and k. By assumption, item k would not be purchased by the customer (even if its value were modified by the adversary). If $p'' < pos_i$, then k could not have been purchased by the customer in position p'', and thus j also could not be purchased in that position because $v'_j < v'_k$. If $p'' > pos_i$, then

item i is purchased before item j would be encountered by the customer in the modified sequence. By performing all such swaps, we recover a modified sequence in which claim 1 holds true. Note that claim 3 also holds true by the same argument.

To show that claim 2 holds true, we use similar mechanics as in the proof that claims 1 and 3 hold true. Let p' be the minimum index in A_2, with item $j \in J_i^1$ again being positioned in slot p', such that item j is attacked by the adversary in the current sequence (e.g., $v'_j > t_{p'+1}$), and where a higher-valued item $k \in J_i^1$ is positioned in slot $p'' > p'$. Consider a modified sequence obtained by swapping the position of items j and k. Item k must be attacked by the adversary in the modified sequence because $v_k > v_j$. Note that item j must still be attacked in the modified sequence as well, because (a) it was attacked in the original sequence in position p', (b) item j is moved to a later slot p'', and (c) by the sequence rule 2–7, we have that $p'' < pos_i$, and the adversary must attack item j to prevent it from being sold before i. Performing all such swaps gives us a sequence that satisfies claim 2. □

Lemma 3. *Consider an optimal sequence of the form 2–7 in which $A_2 = A_3 = \emptyset$. Then there exists an optimal sequence that maximizes $|A_1|$, with items in A_4 being arbitrarily ordered. Moreover, items in A_1 can be ordered by the same greedy approach given in the first claim of Lemma 2.*

Proof. Suppose that item i is a best-profit item that can be sold in any optimal solution, and consider any sequence that results in item i being sold, does not force the adversary to make K attacks on J_i^1 items, and does not maximize the cardinality of A_1. By increasing the number of items in A_1, and retaining the same sequence of items in A_4 (which includes all items in J_i^1), we have that item i is pushed back to a later position in the sequence. Note that no items in A_1 can be purchased by the customer (by construction), and that it is impossible to sell items in A_4 (by assumption). Item i must therefore be sold in this later position. The fact that items in A_4 can be arbitrarily ordered is due to the fact that A_2 is empty, which implies that the adversary has made

fewer than K attacks. Thus, there is no implicit requirement to order items in A_4 that forces the adversary to attack. The justification for the greedy algorithm used to arrange items in A_1 is the same as given in Lemma 2. □

Observe that it is difficult to predict before solving a problem whether an optimal sequence will be generated according to Lemma 2 or 3, and if the former, which value of p $(= |A_1|)$ should be used. Therefore, an algorithm used to solve the case in which our uncertainty set is of the form will consider each possibility allowed by Lemmas 2 and 3.

To state this algorithm, when trying to sell item $i = 1, \ldots, n$ after establishing that items $1, \ldots, i-1$ cannot be sold, we sort items in J_i^1 and J_i^2 in nonincreasing order of their nominal values, to yield the sequences $\alpha(1), \ldots, \alpha(J_i^1)$ and $\beta(1), \ldots, \beta(|J_i^2|)$, respectively. We first attempt to order the items according to Lemma 3. If item i cannot be sold by this sequence, we set an integer parameter $P = |A_1|$ in the sequence produced by Lemma 3, and attempt to create a sequence generated according to Lemma 2 for each $p = 0, \ldots, P$. (Note that there may not be a sequence possible for some values of p: If p is too small, then there may not be enough items in J_i^1 to fill all slots between slot $p+1$ and $pos_{\alpha(K)}$, where the last item in A_2 must be scheduled.) If no sequence can be found that sells item i, then we set $i = i + 1$ and restart the process. The algorithm ends as soon as a sequence is found that sells item i.

Note that this algorithm can be performed in $O(n^3)$ steps, given by the $O(n^2)$ complexity of searching through sequences produced by Lemmas 2 and 3 in trying to sell item i, and the $O(n)$ number of items i that must be explored by the algorithm. We leave for future research the exploration of a more sophisticated algorithm that would attempt to reduce the effort required to search the sequences given by Lemmas 2 and 3.

2.4 Maximization of Expected Profit

In this section, we examine an alternative characterization of uncertainty that arises in the seller's problem. Here, the profit, value, and customer threshold data are all

potentially uncertain. We model this uncertainty by considering a set Q of scenarios, where scenario $q \in Q$ occurs with probability π^q, and in which all profit, value, and threshold data associated with scenario q is superscripted by q (e.g., v_i^q reflects the value of item i in scenario q, and so on). Denote $\pi^q \geq 0$ as the probability of realizing scenario $q \in Q$, where $\sum_{q \in Q} \pi^q = 1$.

Unlike in Section 2.3, we instead examine the problem of *maximizing expected profit* (which we call problem **EXP**), rather than maximizing the minimum profit that could be obtained from a sequence. More formally, problem EXP seeks a sequence of all items, such that the expected profit (given by the summation over all $q \in Q$ of the item's profit that would be sold in scenario q multiplied by π^q) is maximized. The following theorem and its corollary state that once some degree of uncertainty is incorporated to the problem studied in Section 2.2 (whether for the max-min or max-expectation case), the problem can become substantially difficult in general.

Theorem 2.1. *Problem EXP is NP-hard, even when all profit values are deterministic and the customer uses optimal threshold values.*

Proof. The corresponding decision problem, EXPD, is stated as follows: For a given parameter G, does there exist a sequence whose expected profit is at least G? We will show that EXPD is NP-complete, which implies that EXP is NP-hard. Assuming that the customer solves the stopping problem in polynomial time, EXPD clearly belongs to NP: For any given sequence we can easily determine the item that is chosen by the customer in each scenario, compute the expected profit, and check to see if the expected profit is at least G.

Next we show that EXPD is NP-complete by transforming the classical 3SAT problem to an equivalent instance of EXPD. First, we define 3SAT as follows [35].

> Consider a set of clauses C on a set $U = \{u_1, \ldots, u_n\}$ of n binary variables (which can either take values of true or false). Each clause contains three "literal" values, each of which is a true or false value for one particular

variable. Does there exist a "truth assignment" of binary values to the variables in U, i.e., an assignment of true or false values for every variable in U such that every clause has a literal that matches some value in the assignment?

We denote u_i^T (u_i^F) as a literal having a true (false) value for variable i. For instance, if a clause consists of literals $\{u_1^T, u_3^F, u_6^T\}$, then any truth assignment must satisfy the condition that either $u_1 = \texttt{true}$, or $u_3 = \texttt{false}$, or $u_6 = \texttt{true}$.

Consider any 3SAT instance with n variables and m clauses, denoted by $C_j,\ \forall j = \{1, \ldots, m\}$. We define items $T_i,\ \forall i = \{1, \ldots, n\}$, corresponding to u_i^T literals and $F_i,\ \forall i = \{1, \ldots, n\}$, corresponding to u_i^F literals in our EXPD instance. Each of these $2n$ items has a profit of 1. We define one further item, denoted by Z, which has a profit of 0. Let the customer's threshold values be optimal for her, as computed in Equations 2–1a and 2–1b. Also, we define parameter $t^\star$ to be a value satisfying the relationship $t_{n+1} \leq t^\star < t_{n+2}$.

Now let scenario set $Q = \{1, \ldots, n+m\}$, and define the customer values for each scenario as follows:

$$v_Z^q = 100 \qquad \forall q = 1, \ldots, n+m, \tag{2–8a}$$

$$v_{T_q}^q = t^\star \qquad \forall q = 1, \ldots, n, \tag{2–8b}$$

$$v_{F_q}^q = t^\star \qquad \forall q = 1, \ldots, n, \tag{2–8c}$$

$$v_w^q = t^\star \qquad \forall q = n+1, \ldots, n+m, \text{ item } w \text{ corresponds to a literal in } C_{q-n} \tag{2–8d}$$

$$v_i^q = 0 \qquad \text{otherwise.} \tag{2–8e}$$

Note that in 2–8d, the index w refers to the item T_i (F_i), if the literal u_i^T (u_i^F) is a member of the clause C_{q-n}. Finally, $\pi^q = 1/(n+m),\ \forall q \in Q$, and $G = 1$.

To prove that the EXPD instance is equivalent to the 3SAT instance, we show there exists a 3SAT solution if and only if there also exists a solution to EXPD. As a preliminary

note, for any sequence we define *early items* as the first n items in the sequence, and *late items* as the next n items (but not the last item in slot $2n+1$).

First, suppose that there exists a solution to the 3SAT instance. To construct a sequence, we place item F_i in slot i and item T_i in slot $n+i$ if the 3SAT variable u_i is true, $\forall i = 1, \ldots, n$. Otherwise, if u_i is false, we place item T_i in slot i and item F_i in slot $n+i$, $\forall i = 1, \ldots, n$. Item Z is positioned in slot $2n+1$. Note that in any scenario, an early item will not be chosen, because the early item values equal either $t^\star$ or 0, which are less than t_{n+2} and will not be chosen by the customer when they belong to the first n positions of the sequence. If a late item exists having a value of $t^\star$ in scenario q, the customer will buy one such item at a profit of 1 to the seller in scenario q. For scenario $q = 1, \ldots, n$, note that either T_q or F_q is a late item, and has value $t^\star$ in scenario q. For scenario $q = n+1, \ldots, n+m$, one of the three items in clause $q-n$ corresponds to a late item having value $t^\star$, due to the assumption that the late items correspond to a 3SAT truth assignment. In every scenario, a profit of 1 is obtained, and so the expected profit is $G = 1$. Hence, the EXPD instance has a solution.

Next, suppose that there exists a solution to the transformed EXPD instance. Note that this is equivalent to enforcing the condition that item Z is *not* chosen by the customer in any scenario. We will show that the late items in such a sequence correspond to a solution to the 3SAT instance.

Observe that item Z must be placed in slot $2n+1$. If not, some item T_i (or F_i) must be the last item in the sequence. If its complementary item F_i (T_i) is an early item, the customer skips the early item in scenario i and chooses item Z. Otherwise, item F_i (T_i) is a late item, which implies there exists a pair of items T_k and F_k for some k that are both scheduled as early items. This results in item Z being chosen in scenario k. Hence, item Z must be positioned in slot $2n+1$ in our EXPD solution in order to avoid selling item Z in the first n scenarios. Moreover, by the same logic, exactly one of the items T_i or F_i must be a late item, for $i = 1, \ldots, n$.

Using the above observations, we set the variable u_i to true (false), if item T_i (F_i) is a late item. Because Z is not chosen in any scenario $q = n+1, \ldots, n+m$, a late item corresponds to a literal in clause C_j for each $j = 1, \ldots, m$. Therefore, the proposed 3SAT solution is feasible to the 3SAT instance.

Finally, note that the transformation creates a polynomial number of items and scenarios. It is hence a polynomial transformation if $t^\star$ is polynomially representable (i.e., if we require that the number of bits required to represent $t^\star$ is polynomially bounded by n and m). For expediency, we could assume that the customer uses thresholds with precision bounded by a polynomial function of n. Taking $t^\star = (t_{n+1} + t_{n+2})/2$, we have that an encoding of $t^\star$ can also performed using a polynomial number of bits. More generally, even if the customer uses exact thresholds with unlimited precision, we demonstrate in the Appendix A that a value for $t^\star$, encoded using a polynomial number of bits, can be computed. This completes the proof. □

Indeed, a consequence of this theorem is that the general max-min problem (for any arbitrary $\mathcal{U} \neq \emptyset$ with monotone threshold values) must also be NP-hard.

Corollary 1. *The max-min problem is NP-hard for general uncertainty sets $\mathcal{U}$, even when all profit values are deterministic and the customer uses optimal threshold values.*

Proof. We can use the same transformation given for Theorem 2.1, where $\mathcal{U}$ consists of the discrete value sets given above with the threshold values determined by Equations 2–1a and 2–1b. We would then insist on a worst-case profit of 1, which turns out to be identical to requiring that the expected profit equals 1. Thus, the max-min problem with general uncertainty sets is also NP-hard. Observe that we do not make any claims regarding the inclusion of a decision variant of the max-min problem in the class NP, because it is not clear that solving the adversary's problem is generally achievable in polynomial time for general $\mathcal{U}$. □

The implication of Theorem 2.1 is that no polynomial-time solution exists for problem EXP (unless P = NP). We thus provide a mixed-integer programming (MIP) model for this problem, which can be solved by standard MIP techniques [53]. (Indeed, stochastic programming models like the one we face in this section are often solved by decomposition techniques [11], but the development of these algorithms is beyond the scope of this capter.)

To formulate this MIP, we let $N = \{1, \ldots, n\}$ for convenience, and define the following set of decision variables. Let $x_{ij},\ \forall i \in N,\ j \in N$, be a binary decision variable that equals 1 if item i is in slot j of the sequence and 0 otherwise. Also, let $z_j^q,\ \forall j \in N,\ q \in Q$, be a binary decision variable that equals 1 if the item in slot j is chosen by the customer in scenario q, and 0 otherwise. We define a new parameter $a_{ij}^q,\ \forall i \in N,\ j \in N,\ q \in Q$, which equals 1 if item i could possibly be chosen by the customer in scenario q if placed in slot j, i.e.,:

$$a_{ij}^q = \begin{cases} 1 & \text{if } i \in N,\ j \in \{p_i, \ldots, n\},\ q \in Q, \\ 0 & \text{otherwise.} \end{cases} \tag{2–9}$$

We begin by stating a nonlinear MIP that models problem EXP:

$$\max \quad \sum_{q\in Q} \pi^q \sum_{i\in N} b_i \sum_{j\in N} x_{ij} z_j^q \tag{2–10a}$$

$$\text{s.t.} \quad \sum_{j\in N} x_{ij} = 1 \qquad \forall i \in N \tag{2–10b}$$

$$\sum_{i\in N} x_{ij} = 1 \qquad \forall j \in N \tag{2–10c}$$

$$z_j^q \geq a_{ij}^q x_{ij} - \sum_{k=1}^{j-1} z_k^q \qquad \forall i \in N,\ j \in N,\ q \in Q \tag{2–10d}$$

$$z_j^q \leq \sum_{i\in N} a_{ij}^q x_{ij} \qquad \forall j \in N,\ q \in Q \tag{2–10e}$$

$$z_j^q \leq 1 - \sum_{k=1}^{j-1} z_k^q \qquad \forall j \in N,\ q \in Q \tag{2–10f}$$

$$x_{ij} \in \{0,1\} \qquad \forall i \in N,\ j \in N \tag{2–10g}$$

$$z_j^q \in \{0,1\} \qquad \forall j \in N,\ q \in Q. \tag{2–10h}$$

The objective function 2–10a calculates the expected profit: For each scenario $q \in Q$, if $z_j^q = 1$, then the seller receives a profit of b_i, weighted by probability π^q, if item $i \in N$ is placed in slot $j \in N$ (i.e., if $x_{ij} = 1$). Constraints 2–10b and 2–10c guarantee that each item is assigned to exactly one slot and vice versa. Constraints 2–10d–2–10f enforce the condition that z_j^q equals 1 if and only if j is the first slot for which the assigned item i satisfies the condition $v_i^q > t_{j+1}$ in scenario q. To see this, consider an item i positioned in slot j ($x_{ij} = 1$), and observe that if $v_i^q \leq t_{j+1}$, then $a_{ij}^q = 0$. Thus, Constraints 2–10e imply that $z_j^q = 0$ in this case. If however $v_i^q > t_{j+1}$, and thus $a_{ij}^q = 1$, then there are two cases to consider. If there does not exist a slot before j whose corresponding item has been chosen (e.g., $\sum_{k=1}^{j-1} z_k^q = 0$), then Constraints 2–10d force $z_j^q = 1$. If there does exist a $k \in \{1, \ldots, j-1\}$ such that $z_k^q = 1$, then Constraint 2–10f force $z_j^q = 0$ as desired. Observe therefore that Constraints 2–10h can be replaced simply with

$z_j^q \geq 0,\ \forall j \in N,\ q \in Q$, noting that z-variables must be binary-valued given binary x-values.

Although the objective function is nonlinear, it can be easily converted to a linear function due to the facts that (a) nonlinearity only arises due to the nonlinear terms $x_{ij}z_j^q$, and (b) the x-variables are restricted to be binary-valued. By introducing a new set of variables $\psi_{ij}^q,\ \forall i \in N,\ j \in N,\ q \in Q$, which are designed to take on the value of $x_{ij}z_j^q$, we obtain the following linear MIP:

$$\max \quad \sum_{q \in Q} \pi^q \sum_{i \in N} b_i \sum_{j \in N} \psi_{ij}^q \tag{2–11a}$$

$$\text{s.t.} \quad \text{Constraints (2–10b)–(2–10h)}, \tag{2–11b}$$

$$\psi_{ij}^q \leq x_{ij} \qquad \forall i \in N,\ j \in N,\ q \in Q, \tag{2–11c}$$

$$\psi_{ij}^q \leq z_j^q \qquad \forall i \in N,\ j \in N,\ q \in Q. \tag{2–11d}$$

Observe that Constraints 2–11c and 2–11d force $\psi_{ij}^q \leq x_{ij}z_j^q$; equality comes from the fact that optimization will force the ψ-variables to take on their largest permissible values.

CHAPTER 3
A MIXED-INTEGER BILEVEL PROGRAMMING APPROACH FOR A COMPETITIVE PRIORITIZED SET COVERING PROBLEM

3.1 Introduction and Literature Study

We address in this chapter a two-player Stackelberg game on a prioritized set covering problem. In the (one-player) prioritized set covering problem, there exists a set of items that can be selected in order to satisfy a set of clauses. Specifically, each clause contains an ordered partial set of the items. The player incurs costs for each item that is selected, and receives rewards based on satisfied clauses. In particular, a clause is satisfied only by the highest-ranked item (if any) that the player selects, and the reward granted to the player from this clause depends on the item that satisfies the clause.

In the two-player problem we consider, the players act in a Stackelberg leader-follower fashion, in which the follower acts with full knowledge of the items selected by the leader. The leader is aware of the follower's objective, and makes its decisions in anticipation that the follower will optimize its response to the leader's decision. The main contribution that we make in this chapter is an exact solution method for a bilevel programming model representing this game, where the bilevel program involves binary variables representing decisions made by each player.

While the focus of this chapter is on the general two-player prioritized set covering problem, we briefly discuss two scenarios in which this particular problem arises.

- A natural setting for this problem arises in new product development and introduction, which is one of the most important strategic decisions for firms in a competitive market [5, 30, 46, 48]. Here, the set covering items may represent potential products that can be developed by a company, and the clauses may represent customers (or market segments) that wish to purchase products. Each customer has a prioritized list of products that (s)he would buy if available. After the leader firm introduces a set of products, the follower firm (which may actually comprise a set of competing firms), responds by introducing its own set of products. Customers then choose a product that has the most utility to them. The introduction of the follower's products may therefore substantially reduce the profits anticipated by the leader.

This concern is especially relevant in the presence of predatory firms that explicitly seek to minimize the leader's profit. Smith et al. [68] study a two-stage product introduction game similar to the one mentioned above, but in which the follower seeks to minimize the leader's profit. In this case, the leader establishes a product introduction strategy that is robust to any possible actions taken by the follower. However, the predatory model may be far too conservative in some practical settings. The current study, by contrast, focuses on the case in which the follower simply acts to maximize its own profits rather than to minimize the leader's profits. It is worth noting that the algorithm employed in [68] is based on the principle that the leader's objective function is limited by the worst-possible follower's response to the leader's actions. As a result, this algorithm is not valid for the problem considered in this chapter, and the approach taken in the present chapter must be fundamentally different from the one taken in [68].

- Another application area in which this problem arises is in competitive facility location. In this setting, there exists a set of potential facility locations (the set covering items) and geographically located customers (set covering clauses). Customers will gravitate to the most convenient located facility, and hence a customer's preference list is governed by the distance from facility locations. The leader, in making a one-time decision on where to deploy facilities, may thus be concerned about the plans of a competitor in attracting customers across the region under consideration.
  Once again, the presence of a predatory follower has been studied in the literature, based on min-max cutting-plane principles. We refer the reader to [22, 23, 45, 58, 60] for recent research in this field. To our knowledge, though, no research has yet been tailored to the case in which the follower is interested in maximizing its own profit, rather than minimizing the leader's profit. Moreover, the approaches taken in these papers once again cannot be directly extended to the problem under investigation in the present chapter.

Based on the foregoing application areas, we use the more intuitive terms "products" to describe the set covering items, and "customers" to describe the set covering clauses throughout this chapter.

Multilevel programs are mathematical programs in which some of the decision variables are constrained to be optimal with respect to some other mathematical programs [47]. These models often arise in the context of /-level, nonzero-sum games, where the strategy of the player at a specific level is known by the upper-level players. The availability of different decisions to the lower-level players is significantly affected

by the actions taken by the upper-level players [3]. These problems are referred to generally as *mathematical programs with equilibrium constraints* [17, 65].

In particular, bilevel programs are of substantial importance due to the fact that they have been widely applied to Stackelberg game settings that involve two interacting players in different levels, pursuing different objectives, but using a set of common resources. Vicente and Calamai [72] and Dempe [26] conduct literature surveys on the bilevel programs and their applications. A more recent overview of bilevel programming is given by Colson et al. [24].

Linear bilevel programs (i.e., bilevel problems in which both the upper- and lower-level problems are linear programs, given fixed values of the other player's variables) are by far the most studied cases among multilevel programs. Such problems exhibit properties that make them amenable to methods that use a combination of branch-and-bound and implicit search over extreme points. Branch-and-bound can be applied to a 0-1 mixed-integer program converted from the linear bilevel program by substituting the lower-level optimality constraint with an equivalent KKT system [1]. Implicit search methods exploit the fact that an optimal solution of a linear bilevel program will always exist at an extreme point of the constraint set [10].

Candler and Townsley [18] suggest an implicit search method in which a subset of all possible optimal bases for the lower-level problem is examined in order to reach an optimal basic feasible solution to the upper-level problem. Bialas and Karwan [10] report several algorithms to solve linear bilevel programs, including approaches that provably identify local optimal solutions. A study of characterization, solution approaches, and related models for linear bilevel programs have also been conducted by Wen and Hsu [74]. By studying a mixed-integer programming reformulation of the linear bilevel problem, Audet et al. [1] introduce three sets of valid inequalities within a branch-and-cut framework. These inequalities are shown to be valid for all bilevel feasible solutions, while cutting off the solutions that violate the complementarity

constraints. Inexact solution approaches using Genetic Algorithms and Tabu Search methods have been also studied in the literature; see [36] and [41]. The reader is referred to the extensive studies by Bard [3] and Dempe [25] for a thorough treatment of linear bilevel programming methods.

Discrete bilevel programs, however, entail a different set of challenges. Moore and Bard [51] propose a specially tailored branch-and-bound algorithm for mixed-integer bilevel programs. They develop node-fathoming rules to identify the nodes in which the relaxed problem is infeasible, or is not better than the incumbent solution. Utilizing those rules, they develop a branch-and-bound method that computes an optimal solution within a finite number of steps. Another exact solution method suggested by Thirwani and Arora [16, 71] iteratively eliminates optimal solutions to bilevel programming relaxations in a cutting-plane fashion, whenever these solutions are not optimal to the lower-level problem. DeNegre and Ralphs [27] employ a similar approach by solving the same bilevel programming relaxation, but prescribe different cutting planes computed from the constraints that are binding at the relaxation's optimal points. Mitsos [49] derives another family of valid inequalities that can be added to the same relaxations of the mixed-integer bilevel problem, which leads to a finitely convergent algorithm.

More complex forms of bilevel programs have been also studied. Mitsos et al. [50] present a solution method to find a global solution for nonlinear bilevel programs. The convergence proof of their algorithm is based on necessary conditions required for both lower- and upper-level problems. By converting the lower-level problem into a multi-parametric programming problem, Faisca et al. [29] propose another solution method to reach a global solution of bilevel programs with quadratic objective functions. Ozaltin et al. [56] study a bilevel stochastic knapsack problem in which uncertainty is present in the right-hand-side parameters.

The remainder of this chapter is organized as follows. In Section 3.2, we state the formal definition of our problem. We then develop a class of exact solution methods

for the problem based on cutting-plane generation for a so-called high-point problem in Section 3.3. Finally, we report computational results of our proposed algorithm in Section 3.4.

3.2 Problem Formulation

Define set N, with $n = |N|$, as the products in the set covering problem, and let M be the set of customers. Customer $i \in M$ has an ordered product preference list, $O_i = (p_i^1, p_i^2, \ldots, p_i^{k(i)})$, that represents the relative utility of each product to customer i. Customer i will purchase the highest-ranked product among all products in O_i that have been selected by either player, or will purchase no product at all if no item in O_i is available.

The leader starts the game by selecting its set of products. With the knowledge of the leader's decision, the follower then selects its set of products. The revenue earned by the players from customer i is computed as follows. Suppose that customer i purchases product j. If one player selects product j, it earns a revenue of r_{ij}. If both players select product j, the revenue will be divided based on a coefficient $\rho_{ij} \in [0, 1]$, such that the leader earns $\rho_{ij} r_{ij}$, and the follower earns $(1 - \rho_{ij}) r_{ij}$. The leader's (follower's) cost to select product j is given by b_j (c_j). Furthermore, we impose a budget limit B for the leader, which limits the total cost incurred in selecting the leader's products.

Note that the ρ-values discussed above are useful in expanding the scope of problems that can be considered under this framework. If for instance selecting some product $j \in N$ should block the follower from selecting the same product, then setting $\rho_{ij} = 1$ for each $i \in M$ equivalently models this requirement by nullifying any shared revenues that the follower could obtain from repeating the leader's selection of product j.

To model this problem, we define decision variables $x_j = 1$ if the leader selects product $j \in N$, and $x_j = 0$ otherwise. Similarly, define decision variables $y_j = 1$ if the follower selects product $j \in N$, and $y_j = 0$ otherwise. Let w_{ij} and z_{ij} be variables

representing the revenue earned from the sale of product $j \in N$ to customer $i \in M$ by the leader and the follower, respectively. Also, define sets $H_{ij}, \forall i \in M,\ j \in O_i$, as the set of products that have a higher rank than product j in the preference list for customer i. We have the following formulation.

$$\max \quad -b^T x + \sum_{i \in M} \sum_{j \in O_i} w_{ij} \tag{3–1a}$$

$$\text{s.t.} \quad b^T x \leq B \tag{3–1b}$$

$$w_{ij} \leq r_{ij} x_j \qquad \forall i \in M,\ j \in O_i \tag{3–1c}$$

$$w_{ij} \leq r_{ij}(1 - x_k) \qquad \forall i \in M,\ j \in O_i,\ k \in H_{ij} \tag{3–1d}$$

$$w_{ij} \leq r_{ij}(1 - y_k) \qquad \forall i \in M,\ j \in O_i,\ k \in H_{ij} \tag{3–1e}$$

$$w_{ij} \leq r_{ij}(1 - (1 - \rho_{ij}) y_j) \qquad \forall i \in M,\ j \in O_i \tag{3–1f}$$

$$x_j \in \{0, 1\} \qquad \forall j \in N \tag{3–1g}$$

$$y \text{ is part of an optimal solution to (3–2)}, \tag{3–1h}$$

where Problem 3–2 is defined as follows, given the leader's decision variable values, $\bar{x}$:

$$(\text{Problem } F_{\bar{x}}) : \theta_{\bar{x}} = \max \quad -c^T y + \sum_{i \in M} \sum_{j \in O_i} z_{ij} \tag{3–2a}$$

$$\text{s.t.} \quad z_{ij} \leq r_{ij} y_j \qquad \forall i \in M,\ j \in O_i \tag{3–2b}$$

$$z_{ij} \leq r_{ij}(1 - y_k) \qquad \forall i \in M,\ j \in O_i,\ k \in H_{ij} \tag{3–2c}$$

$$z_{ij} \leq r_{ij}(1 - \bar{x}_k) \qquad \forall i \in M,\ j \in O_i,\ k \in H_{ij} \tag{3–2d}$$

$$z_{ij} \leq r_{ij}(1 - \rho_{ij} \bar{x}_j) \qquad \forall i \in M,\ j \in O_i \tag{3–2e}$$

$$y_j \in \{0, 1\} \quad \forall j \in N. \tag{3–2f}$$

The objective function 3–1a represents the leader's profit. The product selection budget is enforced by Constraint 3–1b. We next guarantee that $w_{ij} = 0$ if product j has not been offered by the leader (Constraints 3–1c), if the leader selects at least one product $k \in H_{ij}$ (Constraints 3–1d), or if the follower will select a product $k \in H_{ij}$ (Constraints 3–1e). Finally, Constraints 3–1f state that if product j has been selected by both players, then the leader cannot earn more than $\rho_{ij} r_{ij}$ from selling product j to customer i. Binariness of x-variables is enforced by Constraints 3–1g.

The bilevel nature of the game is enforced by Constraint 3–1h. Note that Constraint 3–1h permits the leader to select any vector y that is an optimal follower's response given the leader's action x. As such, this model is optimistic in that it assumes that the follower breaks ties among alternative optimal solutions in favor of the leader. For the follower problem, the objective function 3–2a and the Constraints 3–2b–3–2e are defined analogously to the leader problem.

In the rest of this chapter, we define $X = \{x \in \{0,1\}^n : b^T x \leq B\}$ as possible decisions the leader can take. We conclude this section by establishing the follower problem's complexity (see Appendix B for the proof).

Theorem 3.1. *Problem $F_{\bar{x}}$ is NP-hard in strong sense.*

An immediate result of Theorem 3.1 is that Problem 3–1 is also NP-hard.

3.3 Exact Solution Method

In this section, we describe an exact solution method tailored for the bilevel problem stated in the previous section. Note that representing Constraint 3–1h via linear inequalities is difficult, because it imposes the optimality of y-variables to a mixed-integer program, which prohibits us from substituting Constraint 3–1h with an equivalent KKT system. Therefore, we suggest a reformulation to Problem 3–1 that is amenable to solution via a cutting-plane algorithm.

Given $\bar{x} \in X$, let $\bar{y}$ be any follower's decision vector. (Note that the optimal w- and z-variables are easily computable given values for x and y; hence, we refer to a

leader/follower solution pair by just (x, y) where convenient.) We call $(\bar{x}, \bar{y})$ a *bilevel feasible* solution to Problem 3–1 if $\bar{y}$ (along with some $\bar{z}$) solves problem $F_{\bar{x}}$. Letting Ω be the set that contains all bilevel feasible solutions of Problem 3–1, we can state the following reformulation of Problem 3–1.

$$\max \quad -b^T x + \sum_{i \in M} \sum_{j \in O_i} w_{ij} \tag{3–3a}$$

$$\text{s.t.} \quad b^T x \leq B \tag{3–3b}$$

$$w \in \mathcal{W}(x, y) \tag{3–3c}$$

$$z \in \mathcal{Z}(x, y) \tag{3–3d}$$

$$x, y \in \{0, 1\}^n \tag{3–3e}$$

$$(x, y) \in \Omega, \tag{3–3f}$$

where $\mathcal{W}(x, y)$ is the polyhedral set that is defined by Constraints 3–1c–3–1f, and $\mathcal{Z}(x, y)$ is the polyhedral set defined by Constraints 3–2b–3–2e.

We define the *high-point* problem (HPP) as the relaxation of Problem 3–3 obtained by omitting Constraints 3–3f. Using the concept introduced by Moore and Bard [51], we can equivalently define HPP as the problem obtained by combining all constraints in Problem 3–1 and 3–2, and discarding Constraint 3–1h. The motivation for defining HPP is to cope with the difficulty of obtaining the explicit definition of the set Ω. Our approach starts by solving HPP, and verifies whether the computed optimal solution of HPP belongs to Ω. If so, this solution must be optimal to 3–3, because HPP is a relaxation of 3–3. Otherwise, we can augment HPP with a cutting plane and re-solve HPP in an iterative fashion. In Section 3.3.1, we formally state this cutting-plane algorithm, and describe auxiliary separation routines in Section 3.3.2 for generating cutting planes within this scheme.

3.3.1 Cutting-Plane Algorithm

Let (x^*, y^*) be (part of) an optimal HPP solution. If y^* is optimal to F_{x^*}, then $(x^*, y^*) \in \Omega$, and hence (x^*, y^*) must be optimal to Problem 3–3. Otherwise, we need to identify a cutting plane, i.e., a valid inequality that is violated by the current bilevel infeasible point (x^*, y^*). We begin by stating a lower bound on θ_{x^*}, for any $x^* \in X$.

Lemma 4. *Let e be a vector of n ones. Then $\theta_e \leq \theta_{x^*}, \ \forall x^* \in X$.*

Proof. Because $x_j^* \leq e_j, \ \forall j \in N$, and $x^* \in X$, F_{x^*} is a relaxation of F_e, which implies that $\theta_e \leq \theta_{x^*}$. □

The next proposition states one class of cutting planes.

Proposition 3.1. *Let (x^*, y^*) be a bilevel infeasible solution, and define $M = \theta_{x^*} - \theta_e$. The following inequality is valid to Problem 3–3, and cuts off (x^*, y^*).*

$$-c^T y + \sum_{i \in M} \sum_{j \in O_i} z_{ij} \geq \theta_{x^*} - M \sum_{j \in N} \left((1 - x_j^*) x_j + x_j^* (1 - x_j) \right) \tag{3–4}$$

Proof. Note that the left-hand-side of 3–4 represents the follower's objective function value. To see that 3–4 is valid, suppose first that $x = x^*$, and that the right-hand-side of 3–4 reduces to θ_{x^*}. In this case, 3–4 simply states that the follower's objective function value must be at least θ_{x^*}, which is valid by our definition of Ω. Also because (x^*, y^*) is bilevel infeasible, $-c^T y^* + \sum_{i \in M} \sum_{j \in O_i} z_{ij}^* < \theta_{x^*}$, and so (x^*, y^*) will be cut off by 3–4. Now suppose that $x \neq x^*$. By choosing $M = \theta_{x^*} - \theta_e$, the right-hand-side of 3–4 is no more than θ_e. The resulting inequality then implies that the follower's objective function value must be at least θ_e, which is valid by Lemma 4. □

The following cutting-plane algorithm, which we call *CPA*, is then given as follows.

Step 0 (Initialization). Set the upper bound $UB = \infty$ and the lower bound $LB = -\infty$.

Define $\mathcal{C}$ (initially empty) as the set of cutting planes that have been added to HPP.

Step 1 (Upper Bound). Identify an optimal solution (x^*, y^*) to the HPP augmented with cutting planes in $\mathcal{C}$, with objective η^*. Set $UB = \eta^*$, and proceed to Step 2.

Step 2 (Lower Bound). Solve F_{x^*}, and obtain an optimal solution $\bar{y}$. Let $\bar{\eta}$ be the optimal objective function value to 3–1 with $x = x^*$ and $y = \bar{y}$. If $\bar{\eta} \geq LB$, set $LB = \bar{\eta}$, and let $(x^*, \bar{y})$ be the incumbent solution. Proceed to Step 3.

Step 3 (Termination/Cut Routine) If $LB = UB$, terminate with the incumbent solution being optimal. Otherwise, add a cutting plane 3–4 to $\mathcal{C}$, and return to Step 1.

Theorem 3.2. *Algorithm CPA identifies an optimal solution to Problem 3–3 in a finite number of iterations.*

Proof. Let ν^* be the optimal objective value to Problem 3–3. Because 3–4 is valid, $\eta^* \geq \nu^*$ after each execution of Step 1 in CPA; also, because $(x^*, \bar{y})$ is feasible to 3–3, $\bar{\eta} \leq \nu^*$ after each execution of Step 2 in CPA. Now suppose by contradiction that CPA does not terminate finitely. Because X is a finite set, CPA would have to encounter some solution $\hat{x} \in X$ multiple times in Step 1 and 2 of the algorithm. At the first time CPA encounters $\hat{x}$, inequality 3–4 is added to $\mathcal{C}$ with respect to $\hat{x}$. However, the proof of Proposition 3.1 implies that if $x = \hat{x}$, the follower vector y must be bilevel feasible in all subsequent iterations of the algorithm. Therefore, letting $(\hat{x}, \hat{y})$ be the optimal HPP solution found the second time CPA encounters $x = \hat{x}$, we must have that $(\hat{x}, \hat{y})$ is bilevel feasible. This implies that $LB = UB$ at Step 3, and the algorithm would have terminated, which leads to a contradiction. This completes the proof. □

CPA might slowly converge to an optimal solution, particularly when the face of conv(Ω) induced by $3 - -4$ only consists of the point $(x^*, \bar{y})$, where x^* was the point used to generate the valid inequality 3–4 and $\bar{y}$ is an optimal solution to F_{x^*}. As a result, HPP may be gradually augmented with a large number of weak cuts, which impairs its solvability. Because the face on conv(Ω) induced by 3–4 is possibly only one point (i.e., a 0-dimensional face), we refer to them as 0-*cuts*.

In order to find stronger cuts that induce faces of at least $q \geq 1$ dimension, which we call "q-cuts", we state the following corollaries of Proposition 3.1.

Corollary 2. *Let $\mathcal{Q} = \{x^1, \dots, x^{(2^q)}\} \subseteq X$, for some $q \geq 1$, such that*

$$\begin{aligned} x^i &\neq x^k && 1 \leq i < k \leq 2^q, \\ x^1_j &= \cdots = x^{(2^q)}_j && q+1 \leq j \leq n. \end{aligned}$$

Define $\mathcal{Q}' = \{(x^i_1, \dots, x^i_q, \theta_{x^i}) : x^i \in \mathcal{Q}\}$, and assume that

$$\sum_{j=1}^{q} \beta_j x^i_j + \theta \geq \alpha, \ \forall (x^i_1, \dots, x^i_q, \theta_{x^i}) \in \mathcal{Q}', \tag{3–6}$$

for some vector $\beta \in \mathbb{R}^q$ and scalar α. Then the following inequality is valid to 3–3:

$$\sum_{j=1}^{q} \beta_j x_j - c^T y + \sum_{i \in M} \sum_{j \in O_i} z_{ij} \geq \alpha - M \sum_{j=q+1}^{n} (1 - x^1_j) x_j - M \sum_{j=q+1}^{n} x^1_j (1 - x_j), \tag{3–7}$$

where $M = \alpha - \sum_{j=1}^{q} \min\{\beta_j, 0\} - \theta_e$.

Proof. We argue that 3–7 is not violated by any bilevel feasible point. First, consider all bilevel feasible points $(x, y),\ x \in \mathcal{Q}$. In this case, 3–7 reduces to 3–6, which is valid by assumption. Now consider any bilevel feasible point $(x, y),\ x \notin \mathcal{Q}$. In this case, $x_j \neq x^1_j$ for some $j = q+1, \dots, n$. By choosing $M = \alpha - \sum_{j=1}^{q} \min\{\beta_j, 0\} - \theta_e$, the right-hand-side of 3–7 is no more than $\sum_{j=1}^{q} \min\{\beta_j, 0\} + \theta_e$. This implies that 3–7 is not violated because $\beta_j x_j \geq \min\{\beta_j, 0\},\ \forall j = 1, \dots, q$, and $-c^T y + \sum_{i \in M} \sum_{j \in O_i} z_{ij} \geq \theta_e$ from Lemma 4. □

Corollary 3. *Let $(\bar{x}, \bar{y}) : \bar{x} \in \mathcal{Q}$ be a bilevel infeasible solution. If $\sum_{j=1}^{q} \beta_j \bar{x}_j + \theta_{\bar{x}} = \alpha$ for the coefficients that are used to generate 3–7, then 3–7 cuts off $(\bar{x}, \bar{y})$.*

Proof. Let $\bar{z}$ be the value of z-variables corresponding to $(\bar{x}, \bar{y})$. Since $(\bar{x}, \bar{y})$ is bilevel infeasible, we have:

$$\begin{aligned} \sum_{j=1}^{q} \beta_j \bar{x}_j - c^T \bar{y} + \sum_{i \in M} \sum_{j \in O_i} \bar{z}_{ij} &< \sum_{j=1}^{q} \beta_j \bar{x}_j + \theta_{\bar{x}} \\ &= \alpha \\ &= \alpha - M \sum_{j=q+1}^{n} (1 - x^1_j) \bar{x}_j - M \sum_{j=q+1}^{n} x^1_j (1 - \bar{x}_j), \end{aligned}$$

where the last equality is true because $\bar{x} \in \mathcal{Q}$. □

Our motivation for using the term q-cut stems from the fact that if the inequality 3–6 is binding for at least $q+1$ points in $\mathcal{Q}'$, then the resulting inequality 3–7 must be binding on at least $q+1$ affinely independent bilevel feasible points, implying that it induces a $q+1$ (or higher) dimensional face of conv(Ω). As a result, Corollaries 2 and 3 permit us to obtain stronger cuts within CPA (which remains correct and finitely convergent by the same argument in the proof of Theorem 3.2). For instance, let (x^1, y^1) be a bilevel infeasible solution obtained by solving HPP, and suppose that we seek a 2-cut based on this solution. We start by constructing set $\mathcal{Q}'$ from Corollary 2 for $q = 2$, which contains $(x_1^i, x_2^i, \theta_{x^i}),\ \forall i = 1, \ldots, 4$. Next we find any inequality $\beta_1 x_1 + \beta_2 x_2 + \theta \geq \alpha$ that is binding at $(x_1^1, x_2^1, \theta_{x^1})$ and two of the other three points, and is valid with respect to the remaining point in $\mathcal{Q}'$. This inequality satisfies the necessary assumptions for Corollaries 2 and 3, and hence, a 2-cut is generated of the form 3–7 based on α, β_1, and β_2.

3.3.2 Follower and Separation Subproblem

In this section, we present an alternative approach to generate a $(q+1)$-cut, given some starting q-cut that is known to be valid. The motivation for this approach stems from the observation that employing Corollary 2 requires excessive computational effort for larger values of q. For example, to obtain a 3-cut that is violated by (x^1, y^1), we must identify an inequality in $\mathbb{R}^4$ that is binding at $(x_1^1, x_2^1, x_3^1, \theta_{x^1})$ and three points of the set $\mathcal{Q}' = \{(x_1^i, x_2^i, x_3^i, \theta_{x^i}) : i = 2, \ldots, 8\}$, and is valid with respect to the other four points in $\mathcal{Q}'$. This requires the solution of eight instances of the follower problem. In addition, we may also have to examine $\binom{7}{3}$ possible hyperplanes in the worst case to obtain an inequality satisfying the conditions of Corollary 2.

Given the initial q-cut, and a set of points $\{x^1, \ldots, x^{q+1}\} \subset \mathcal{Q}$ binding on this inequality (where $\mathcal{Q}$ is defined as in Corollary 2), our strategy constructs a single new

point $\tilde{x}$ as follows.

$$\begin{aligned} \tilde{x}_j &= 1 - x_j^1 \qquad && j = q+1 \\ \tilde{x}_j &= x_j^1 \qquad && j \neq q+1,\ j \in \{1, \ldots, n\} \end{aligned} \tag{3–9}$$

It is easy to verify that $\tilde{x}$ remains affinely independent from the initial set of $q+1$ binding points. Denoting the inequality that is binding at these $q+2$ points by $\sum_{j=1}^{q+1} \beta_j x_j + \theta \geq \alpha$, we have a candidate for 3–6 that can be potentially used to obtain a $(q+1)$-cut. However, inequality 3–7 generated based on (α, β) may not be valid for all x-vectors. Therefore, we seek an efficient way of verifying the validity of the $(q+1)$-cut. Consider the following separation problem:

$$\min \quad {\beta'}^T x + \theta_x - \alpha' \tag{3–10a}$$

$$\text{s.t.} \quad x \in X, \tag{3–10b}$$

where β' is the coefficient vector of x-variables in 3–7 based on β, α' is the constant term of 3–7, and θ_x is defined as earlier. A solution $x \in X$ has a negative objective function value to problem 3–10 if and only if it violates inequality 3–7 generated according to (α', β').

However, note that problem 3–10 is a two-stage mixed-integer program, because computing θ_x requires solving F_x, which is embedded in 3–10 as an inner optimization problem. Hence, 3–10 is difficult to solve due to the nonconvex inner problem. Our strategy is to substitute the inner problem with a convex *restriction* (which yields a *relaxation* of 3–10). If the relaxed problem has a nonnegative value for the objective function at optimality, then the proposed $(q+1)$-cut must be valid. Otherwise, if the relaxed version of 3–10 has a negative optimal objective function value, the $(q+1)$-cut may or may not be valid. The merit of using a *convex* restriction for the inner problem in 3–10 is that it can be substituted with its dual, allowing us to state the relaxation of 3–10 as one mixed-integer program. We seek a relaxation of $3--10$ whose optimal objective

value is not much smaller than the optimal objective value of 3–10, in order to verify that the generated inequality is valid.

Before describing our relaxations of 3–10, we summarize by substituting Step 3 with the following two steps.

Step 3a (Termination) If $LB = UB$, terminate with the incumbent solution being optimal. Otherwise, generate a cutting plane 3–7 for some value of q and proceed to Step 3b.

Step 3b (Cut Routine) Create a new point using 3–9, generate a new candidate inequality, and verify the validity of the inequality by solving a relaxation of 3–10. If the objective of 3–10 is negative, add the q-cut to $\mathcal{C}$, and return to Step 1. Otherwise, store the newly-generated $(q+1)$-cut as the last identified cut, increment q by one, and repeat Step 3b.

The following four subsections consider alternative convex restrictions of the inner problem, and the resulting mathematical program for the separation problem relaxation.

3.3.2.1 The "same action" restriction

Consider the setting in which the follower is restricted to select the same set of products selected by the leader. Then for a given x, the follower problem now becomes:

$$-c^T x + \max \sum_{i \in M} \sum_{j \in O_i} z_{ij} \tag{3–11a}$$

$$\text{s.t. } z_{ij} \le r_{ij} x_j \qquad \forall i \in M,\ j \in O_i \tag{3–11b}$$

$$z_{ij} \le r_{ij}(1 - \rho_{ij} x_j) \qquad \forall i \in M,\ j \in O_i \tag{3–11c}$$

$$z_{ij} \le r_{ij}(1 - x_k) \qquad \forall i \in M,\ j \in O_i,\ k \in H_{ij}. \tag{3–11d}$$

Let $\mu_{ij}, \tau_{ij},\ \forall i \in M,\ j \in O_i$, be the dual variables associated with constraints 3–11b and 3–11c, respectively. Also, let $\pi_{ijk},\ \forall i \in M,\ j \in O_i,\ k \in H_{ij}$, be the dual variables associated with constraints 3–11d. By substituting the follower problem with the dual of

3–11 and combining with $3 - -10$, the leader problem becomes:

$$\min \ (\beta - c)^T x + \sum_{i \in M} \sum_{j \in O_i} \left(r_{ij} x_j \mu_{ij} + r_{ij}(1 - \rho_{ij} x_j)\tau_{ij} + \sum_{k \in H_{ij}} r_{ij}(1 - x_k)\pi_{ijk} \right) - \alpha \tag{3–12a}$$

$$\text{s.t.} \ \ \mu_{ij} + \tau_{ij} + \sum_{k \in H_{ij}} \pi_{ijk} = 1 \qquad \forall i \in M, \ j \in O_i \tag{3–12b}$$

$$\mu_{ij}, \tau_{ij} \geq 0 \qquad \forall i \in M, \ j \in O_i \tag{3–12c}$$

$$\pi_{ijk} \geq 0 \qquad \forall i \in M, \ j \in O_i, \ k \in H_{ij} \tag{3–12d}$$

$$x \in X. \tag{3–12e}$$

Problem 3–12 has quadratic terms $\tau_{ij} x_j$ and $\pi_{ijk} x_k$. We present a generic set of inequalities (introduced in [40]) that serve to linearize these terms. Given a binary variable δ and a continuous variable $\sigma \in [0, r]$, we replace $t = \delta\sigma$ and restrict t via the polyhedral set:

$$\mathcal{P}_1(\delta, \sigma, r) = \{t \in \mathbb{R}^+ : \ t \leq r\delta, \ t \leq \sigma, \ t \geq \sigma + r\delta - r\}. \tag{3–13}$$

We use the inequalities defining this polyhedron to introduce variables $\lambda^1_{ij} = x_j \mu_{ij}$, $\lambda^2_{ij} = x_j \tau_{ij}$, and $\lambda^3_{ijk} = (1 - x_k)\pi_{ijk}$, which linearize all quadratic terms in 3–12a.

$$\min \ (\beta - c)^T x + \sum_{i \in M} \sum_{j \in O_i} \left(r_{ij} \lambda^1_{ij} + r_{ij} \tau_{ij} - \rho_{ij} r_{ij} \lambda^2_{ij} + \sum_{k \in H_{ij}} r_{ij} \lambda^3_{ijk} \right) - \alpha \tag{3–14a}$$

$$\text{s.t.} \ \ \mu_{ij} + \tau_{ij} + \sum_{k \in H_{ij}} \pi_{ijk} = 1 \qquad \forall i \in M, \ j \in O_i \tag{3–14b}$$

$$\lambda^1_{ij} \in \mathcal{P}_1(x_j, \mu_{ij}, 1) \qquad \forall i \in M, \ j \in O_i \tag{3–14c}$$

$$\lambda^2_{ij} \in \mathcal{P}_1(x_j, \tau_{ij}, 1) \qquad \forall i \in M, \ j \in O_i \tag{3–14d}$$

$$\lambda^3_{ijk} \in \mathcal{P}_1(1 - x_k, \pi_{ijk}, 1) \qquad \forall i \in M, \ j \in O_i, \ k \in H_{ij} \tag{3–14e}$$

$$\mu_{ij}, \tau_{ij} \geq 0 \qquad \forall i \in M, \ j \in O_i \tag{3–14f}$$

$$\pi_{ijk} \geq 0 \qquad \forall i \in M, \ j \in O_i, \ k \in H_{ij} \tag{3–14g}$$

$$x \in X. \tag{3–14h}$$

3.3.2.2 The "single product" restriction

Consider an alternative restriction that limits the follower to choose at most one product. We define:

$$\bar{c}_j = -c_j + \sum_{i \in M} \left(r_{ij}(1 - \rho_{ij} x_j) \prod_{k \in H_{ij}} (1 - x_k) \right), \ \forall j \in N. \tag{3–15}$$

Note that $\bar{c}_j$ is the leader's profit associated with product j for a given leader's decision vector $x \in X$. The restricted follower problem is stated as follows.

$$\max \quad \sum_{j \in N} \bar{c}_j y_j \tag{3–16a}$$

$$\text{s.t.} \quad \sum_{j \in N} y_j \leq 1 \tag{3–16b}$$

$$y_j \geq 0 \quad \forall j \in N. \tag{3–16c}$$

By introducing u as the dual variable associated with constraint 3–16b and substituting the inner problem of 3–10 with the dual of 3–16, problem 3–10 becomes:

$$\min \quad \beta^T x + u - \alpha \tag{3–17a}$$

$$\text{s.t.} \quad u \geq \bar{c}_j \quad \forall j \in N \tag{3–17b}$$

$$u \geq 0 \tag{3–17c}$$

$$x \in X. \tag{3–17d}$$

To linearize problem 3–17, let $\mathcal{V} = \{v^1, \ldots, v^{|\mathcal{V}|}\}$ be a set of binary variables. We define the following polyhedral set, which enforces $t = \prod_{i=1}^{|\mathcal{V}|} v^i$:

$$\begin{aligned} \mathcal{P}_2(\mathcal{V}) = \{ & t \in \mathbb{R}^+ : t \leq v, \ \forall v \in \mathcal{V}, \\ & t + |\mathcal{V}| - 1 \geq \sum_{i=1}^{|\mathcal{V}|} v^i \}. \end{aligned} \tag{3–18}$$

Define $\mathcal{V}_{ij} = \{1 - x_k : k \in H_{ij}\},\ \forall i \in M,\ j \in O_i$, and let $\lambda^4_{ij} = \prod_{k \in H_{ij}}(1 - x_k)$. Also, let $\lambda^5_{ij} = x_j \lambda^4_{ij}$. We then obtain the following relaxation of 3–10.

$$\min \quad \beta^T x + u - \alpha, \tag{3–19a}$$

$$\text{s.t.} \quad u - \sum_{i \in M} \left(r_{ij} \lambda^4_{ij} - r_{ij} \rho_{ij} \lambda^5_{ij} \right) \geq -c_j \quad \forall j \in N \tag{3–19b}$$

$$\lambda^4_{ij} \in \mathcal{P}_2(\mathcal{V}_{ij}) \qquad \forall i \in M,\ j \in O_i \tag{3–19c}$$

$$\lambda^5_{ij} \in \mathcal{P}_1(x_j, \lambda^4_{ij}, 1) \qquad \forall i \in M,\ j \in O_i \tag{3–19d}$$

$$u \geq 0 \tag{3–19e}$$

$$x \in X. \tag{3–19f}$$

3.3.2.3 The "same action with one fewer product" restriction

In this restriction, the follower is constrained to select all but at most one of the products offered by the leader, and none that have not already been selected by the leader. Hence, we define:

$$\hat{c}_j = c_j - \sum_{i \in M} \left(r_{ij}(1 - \rho_{ij} x_j) \prod_{k \in H_{ij}} (1 - x_k) \right), \ \forall j \in N, \tag{3–20}$$

as the difference in the follower's profit if they select all of the leader's products except for j, and the profit if they select all of the leader's products. Then, given $x \in X$, the following is a restriction of the follower problem.

$$-c^T x \ + \max \ \sum_{i \in M} \sum_{j \in O_i} z_{ij} + \sum_{j \in N} \hat{c}_j y_j \tag{3–21a}$$

$$\text{s.t.} \ \text{Constraints (3–11b)–(3–11d)} \tag{3–21b}$$

$$\sum_{j \in N} y_j \leq 1 \tag{3–21c}$$

$$y_j \leq x_j \qquad \forall j \in N \tag{3–21d}$$

$$y_j \geq 0 \qquad \forall j \in N. \tag{3–21e}$$

Problem 3–21 can be separated into two subproblems given x: one in terms of z-variables (referred to as the z-subproblem), and the other one in terms of y-variables (referred to as the y-subproblem). Note that the z-subproblem is exactly the same problem as 3–11 (excluding the term $-c^T x$). Let u and $v_j,\ j \in N$, be the dual variables associated with Constraints 3–21c and 3–21d, respectively. We obtain a mixed-integer program by dualizing the z-subproblem similar to Problem 3–12, and dualizing the y-subproblem using u- and v-variables, which results in presence of quadratic terms $x_j v_j$. In order to linearize these terms, we need upper bounds on the optimal values of v-variables. Observe that the dual of the y-subproblem is as follows.

$$\min \quad u + \sum_{j \in N} x_j v_j \tag{3–22a}$$

$$\text{s.t.} \quad u + v_j \geq \hat{c}_j \qquad \forall j \in N \tag{3–22b}$$

$$u \geq 0 \tag{3–22c}$$

$$v_j \geq 0 \qquad \forall j \in N. \tag{3–22d}$$

We claim that in any optimal solution (u^*, v^*), $v_j^* \leq \max\{0, \hat{c}_j\},\ \forall j \in N$. To see this, consider any solution (u^*, v^*) with $v_j^* > \hat{c}_j$ for some $j \in N$. Let $(u^*, \bar{v})$ be the solution obtained by letting $\bar{v}_j = \min\{v_j^*, \hat{c}_j\}$ if $\hat{c}_j \geq 0$, and $\bar{v}_j = 0$ if $\hat{c}_j < 0$. It is easy to verify that $(u^*, \bar{v})$ remains feasible, because $u \geq 0$. Moreover, it yields an objective function value no worse than the value computed from solution (u^*, v^*), because $x \geq 0$ and $u \geq 0$. Thus, $v_j^* \leq \max\{0, \hat{c}_j\},\ \forall j \in N$, which implies $v_j^* \leq c_j$ from 3–20. Using this result, we obtain the following relaxation of Problem 3–10.

$$\min \quad \varphi + u + \sum_{j \in N} \psi_j^1 \tag{3–23a}$$

$$\text{s.t.} \quad u + v_j + \sum_{i \in M} \left(r_{ij} \lambda_{ij}^4 - r_{ij} \rho_{ij} \lambda_{ij}^5 \right) \geq c_j \qquad \forall j \in N \tag{3–23b}$$

$$\psi_j^1 \in \mathcal{P}_1(x_j, v_j, c_j) \qquad \forall j \in N \tag{3–23c}$$

$$u \geq 0 \tag{3–23d}$$

$$v_j \geq 0 \qquad \forall j \in N \tag{3–23e}$$

$$\text{Constraints (3–14b)–(3–14h), (3–19c), (3–19d),} \tag{3–23f}$$

where ψ^1-variables are introduced to linearize the quadratic terms $v_j x_j$, φ is defined as the objective function of Problem 3–14, and variables λ_{ij}^4 and λ_{ij}^5 are defined similar to Problem 3–14.

3.3.2.4 The "same action with one more product" restriction

The fourth restriction restricts the follower to offer all of the products selected by the leader, while having the option of selecting one additional product. For a given vector $x \in X$, the follower solves:

$$-c^T x \; + \max \quad \sum_{i \in M} \sum_{j \in O_i} z_{ij} + \sum_{j \in N} \bar{c}_j y_j \tag{3–24a}$$

$$\text{s.t.} \quad \text{Constraints (3–11b)–(3–11d)} \tag{3–24b}$$

$$\sum_{j \in N} y_j \leq 1 \tag{3–24c}$$

$$y_j \leq 1 - x_j \qquad \forall j \in N \tag{3–24d}$$

$$y_j \geq 0 \qquad \forall j \in N. \tag{3–24e}$$

Let u and v_j, $\forall j \in N$, be the dual variables associated with Constraints 3–24c and 3–24d, respectively. Using an analysis similar to Problem 3–21, and linearizing the

nonlinear terms, we obtain the following relaxation of Problem 3–10.

$$\min \quad \varphi + u + \sum_{j \in N} \psi_j^2 \tag{3–25a}$$

$$\text{s.t.} \quad u + v_j - \sum_{i \in M} \left(r_{ij} \lambda_{ij}^4 - r_{ij} \rho_{ij} \lambda_{ij}^5 \right) \geq -c_j \qquad \forall j \in N \tag{3–25b}$$

$$u \geq 0 \tag{3–25c}$$

$$v_j \geq 0 \qquad \forall j \in N \tag{3–25d}$$

$$\psi_j^2 \in \mathcal{P}_1(1 - x_j, v_j, c_j) \qquad \forall j \in N \tag{3–25e}$$

$$\text{Constraints (3–14b)–(3–14h), (3–19c), (3–19d),} \tag{3–25f}$$

where ψ^2-variables are introduced to linearize the quadratic terms $v_j(1 - x_j)$, φ is defined as the objective function of Problem 3–14, and variables λ_{ij}^4 and λ_{ij}^5 are defined similar to 3–14.

3.4 Computational Results

In this section, we examine the efficacy of our approach on randomly generated test instances. We start by describing different implementation details of the algorithms presented in this chapter, and then show how to tailor the algorithm presented in [49] to our problem. We conduct different computational studies with respect to the product introduction and facility location applications described in Section 3.1.

3.4.1 Implementation Details and Instance Generation

The first three implementations of CPA that we examine, denoted by CPA1, CPA2, and CPA3, do not use any of the separation procedures presented in Section 3.3.2. For CPA1, we implement CPA with $q = 2$. CPA2 generates a 2-cut (and its corresponding $\mathcal{Q}'$), and then attempts to compute a 3-cut by constructing a new point according to 3–9. CPA2 then verifies the validity of the inequality for all points in $\mathcal{Q}'$, which now contains eight points corresponding to 3-cut. If the candidate inequality is valid, it is added to $\mathcal{C}$; otherwise CPA2 augments $\mathcal{C}$ by the initially generated 2-cut. CPA3 is implemented

similarly. It starts with a 3-cut, aiming to convert that inequality to a 4-cut using the same process as given in CPA2.

We also study four implementations of CPA augmented by the separation problems presented in Section 3.3.2. We denote the augmented CPA implementations by ACPA1, ACPA2, ACPA3, and ACPA4, corresponding to the four separation subproblems given in Section 3.3.2. For each implementation, Step 3a in the augmented CPA starts with a 2-cut. Using 3–9, a new candidate inequality is identified and the validity of the inequality is verified via the corresponding separation subproblem.

Finally, we also examine the efficacy of using a hybrid strategy, denoted by HYB, which first identifies a 2-cut and employs Problem 3–10 to obtain a 3-cut. If HYB cannot verify the validity of the candidate inequality by solving Problem 3–10, it explicitly tests the validity of the candidate inequality for all points in the corresponding $\mathcal{Q}'$. If the candidate inequality is not valid, the initial 2-cut is added to $\mathcal{C}$. Otherwise, the HYB approach sets $q = 3$, and executes Step 3b of the augmented CPA.

Finally, we explain how to implement the proposed cutting-plane algorithm by Mitsos [49], denoted by MITS, in order to compare the performance of our algorithm to the existing work in the literature. As a cutting-plane algorithm, MITS is designed to calculate a global optimal solution to *general* nonlinear mixed-integer bilevel programs. To obtain upper bounds (for maximization problems), MITS solves HPP augmented with cutting planes that are different from those proposed in our approach. More precisely, let $(x^k, \bar{y}^k)$ be a bilevel infeasible optimal solution of HPP (possibly augmented with valid inequalities) at iteration $k \geq 1$, and let y^k solve F_{x^k}. MITS seeks a new set, denoted by X^k, such that solutions $(x, y^k),\ \forall x \in X^k$, remain feasible to F_x. Letting f^F be the objective function of 3–2, the following is a valid inequality for Problem 3–3:

$$f^F(x, y) \geq f^F(x, y^k),\ \forall x \in X^k.$$

For the product introduction instances, we generated four test sets, denoted by S1, S2, S3, and S4 by varying $|M| \in \{6, 9\}$ and $|N| \in \{12, 15\}$. Each test set contains ten randomly generated instances. Table 3-1 shows the parameters and the corresponding lower and upper bounds for each parameter. All parameters are randomly generated as integers drawn from a uniform distribution over the stated ranges, except for the ρ-values, which are uniformly generated over the continuous interval [0.1, 0.9]. Note that in Table 3-1, $B_{\min} = 0.75S_b/|N|$ and $B_{\max} = 1.25S_b/|N|$, where S_b is the sum of all generated b-values.

Table 3-1. Product introduction: parameters used to generate test instances

Parameter Name	**Value**
Leader's budget (B)	$[B_{\min}, B_{\max}]$
Leader's product selection costs (b_j)	[70, 190]
Follower's product selection costs (c_j)	[90, 170]
Customer preference list sizes ($\|O_i\|$)	[1, n]
Revenues (r_{ij})	[20, 130]
Revenue share coefficients (ρ_{ij})	[0.1, 0.9]

For the facility location application, we generated test sets S5, S6, S7, and S8 by varying $|M| \in \{12, 15\}$ and $|N| \in \{9, 12\}$. The parameters are reported in Table 3-3. We first randomly generated $|M|$ as the number of customers and $|N|$ as the number of potential facility locations within a rectangle of length 120 and width of 90. Next, based on some randomly generated distance threshold, d, we determined the potential facilities that can serve each customer. For each customer i, we then place all potential facility locations in O_i in nondecreasing order of their distance from customer i, omitting those whose distance from i exceeds d. Note that $r_{ij} = r_{ik}$ for all $j, k \in N$ for this application, because the demand from each customer is assumed to be independent of the selected facility. Finally, $B_{\min}$ and $B_{\max}$ are defined in a similar way to Table 3-1.

We implemented all variants in Visual C++ 8.0 equipped with CPLEX 12.2 Concert Technology on an Intel Core i5 PC with 4 GB of memory. For each implementation, we set the maximum allowable running time to be 1200 seconds.

Table 3-2. Facility location: parameters used to generate test instances

Parameter Name	**Value**
Leader's budget (B)	$[B_{\min}, B_{\max}]$
Leader's facility location selection costs (b_j)	[50, 200]
Follower's facility location selection costs (c_j)	[10, 100]
Distance threshold for customers (d)	[50, 70]
Revenues (r_{ij})	[30, 150]
Revenue share coefficients (ρ_{ij})	[0.1, 0.9]

3.4.2 Results

We now discuss the performance of the CPA variants on the two featured applications. Table 3-3 illustrates the average time and number of cuts necessary for each CPA variant to reach an optimal solution. According to Table 3-3, CPA1 is more efficient than CPA2 and CPA3 in solving instances of product introduction application. On the other hand, CPA1 is outperformed by CPA2 and CPA3 for the facility location application, particularly on larger instances. Note that CPA1 requires more cuts to reach an optimal solution in comparison to CPA2 and CPA3. This is due to the fact that CPA2 and CPA3 are capable of generating stronger cuts. The result shows that for the facility location application, generating stronger cuts are beneficial in that the overall algorithm takes less time to reach an optimal solution. However, the reduction in time required to solve HPP for product introduction instances does not compensate for the extra time required by CPA2 and CPA3 to generate stronger cuts.

Table 3-3. Comparison of CPA implementations

		CPA1		CPA2		CPA3	
Application	Set	Avg Time	Avg Cuts	Avg Time	Avg Cuts	Avg Time	Avg Cuts
Product introduction	S1: ($\|N\|$=12, $\|M\|$=6)	7.5	46	8.92	39	11.02	34
	S2: ($\|N\|$=12, $\|M\|$=9)	20.45	131	23.77	109	24.03	97
	S3: ($\|N\|$=15, $\|M\|$=6)	14.91	109	12.41	76	15.25	66
	S4: ($\|N\|$=15, $\|M\|$=9)	40.95	275	42.01	225	45.34	201
Facility location	S5: ($\|N\|$=9, $\|M\|$=12)	7.34	48	6.68	31	6.48	26
	S6: ($\|N\|$=12, $\|M\|$=12)	101.07	381	90.91	250	89.82	207
	S7: ($\|N\|$=9, $\|M\|$=15)	9.81	46	9.34	30	9.43	26
	S8: ($\|N\|$=12, $\|M\|$=15)	149.36	419	131.37	262	133.79	214

Similarly, we investigate the efficacy of the four augmented CPA implementations. Table 3-4 indicates that for both applications, ACPA2 outperforms ACPA1, ACPA3,

and ACPA4, but not the best available CPA variants. The separation subproblems tend to be substantially difficult to solve, and they either fail to generate a stronger cut, or the generated cut is not strong enough to offset the time spent solving the separation subproblem. Note that for the product introduction instances, ACPA2 employs slightly fewer cuts compared to other implementations, whereas for the facility location application, ACPA4 requires the fewest number of cuts to reach an optimal solution. In fact, ACPA4 computes stronger cuts at the expense of spending extra time to solve more follower problem instances or harder separation problems, which results in longer computation times for the overall algorithm. It is also worth noting that the difference between the performance of ACPA3 and ACPA4 is indistinguishable on the product introduction test instances, but the advantage of using ACPA4 instead of ACPA3 is strongly pronounced on the facility location instances.

Table 3-4. Comparison of augmented CPA implementations

	ACPA1		ACPA2		ACPA3		ACPA4	
Set	Avg Time	Avg Cuts	Avg Time	Avg Cuts	Avg Time	Avg Cuts	Avg Time	Avg Cuts
S1	20.44	41	12.51	30	19.72	41	19.58	41
S2	68.49	115	37.81	101	69.32	115	69	115
S3	41.75	89	26.63	89	56.20	89	53.17	89
S4	152.48	244	78.93	242	251.27	244	241.06	251
S5	20.19	40	12.42	40	26	40	23.01	27
S6	278.86	312	143.97	261	370.18	312	295.81	217
S7	25.33	38	16.14	37	33.83	38	26.63	19
S8	307.89	331	226.67	331	379.23	331	372.50	259

As a result, relaxation 3–19 outperforms the other proposed relaxations in verifying the generated candidate valid inequalities. The results also show that the less restricted versions of 3–11 presented in Sections 3.3.2.3 and 3.3.2.4 do not yield more promising relaxations than 3–14 in general.

Based on the results from Tables 3-3 and 3-4, we also examine the efficacy of HYB. In particular, we employ relaxation 3–19 in the implementation of HYB due to the fact that it turned out to be the best proposed relaxation in Section 3.3.2. We report the results of employing HYB and MITS in Table 3-5, alongside the results from using the

best CPA variant for each application (CPA1 for product introduction and CPA2 for facility location) and the results of ACPA2 previously stated in Tables 3-3 and 3-4, respectively.

Table 3-5. Performance comparison for the best CPA, ACPA2, HYB, and MITS

	Best CPA		ACPA2		HYB		MITS		
Set	Avg Time	Avg Cuts	Avg Time	Avg Cuts	Avg Time	Avg Cuts	Avg Time	Avg Cuts	# Opt
S1	7.5	46	12.51	30	8.74	23	44.72	59	10
S2	20.45	131	37.81	101	26.35	81	497.38	179	8
S3	14.91	109	26.63	89	15.98	63	545.77	159	8
S4	40.95	275	78.93	242	51.83	196	988.74	279	3
S5	6.68	31	12.42	40	6.99	26	37.88	41	10
S6	90.91	250	143.97	261	89.34	185	569.82	154	6
S7	9.34	30	16.14	37	9.94	25	105.53	63	10
S8	131.37	262	226.67	331	134.73	214	893.98	238	4

Note that MITS consumes a large amount of computation time for larger instances, and may fail to reach an optimal solution within the maximum allowable running time. Therefore, we have reported the number of times that MITS terminates within 1200 seconds in the column "# Opt" in Table 3-5. (A CPU time of 1200 seconds is recorded for those instances that do not terminate within the time limit, and the number of cuts generated for MITS within this time limit is factored into the "Avg Cuts" column.)

Based on the results in Table 3-5, MITS is always outperformed by all variants of our proposed algorithm. Note that on product introduction instances, CPA1 remains the best variant of our proposed algorithm. However, the difference between CPA2 and HYB for the facility location test instances is indistinguishable. The merit of using HYB is generally observable in fewer cuts necessary to solve the problem, but this may happen at the expense of spending extra time on solving more follower problem instances or separation problems that may not be beneficial overall (see results of using CPA2 and HYB for sets S2 and S4).

CHAPTER 4
A CUTTING-PLANE ALGORITHM FOR SOLVING A WEIGHTED INFLUENCE INTERDICTION PROBLEM

4.1 Introduction and Literature Study

We consider a scenario in which two players, a defender and an attacker, compete on a directed network $G(V, A)$, where V is the set of nodes and A is the set of arcs. Initially, the defender owns every node in the network, and can protect a subset of nodes against an impending action by the attacker. The attacker then acts, with full knowledge of the defender's action, to capture a set of unprotected nodes. For consistency with prior related studies, we say that captured nodes have been *influenced* by the attacker. This initial action takes place at time 0, and the game continues for T (discrete) time periods according to the following rules.

1. An influenced node remains influenced for the remainder of the time horizon.
2. A node that was protected by the defender cannot be influenced at any time.
3. Consider an unprotected node $j \in V$ that is not influenced at time $t \in \{0, \ldots, T-1\}$. Then node $j \in V$ becomes influenced at time $t+1$ if and only if there are some Q nodes $i \in V$ such that i is influenced at time t, and $(i, j) \in A$.
4. The attacker earns a reward of r_i^t if node i is influenced at time t (but not at time $t-1$, if $t \geq 1$).

The goal of the defender is to minimize the maximum sum of rewards that the attacker can earn (e.g., minimizing the maximum amount of damage that the attacker could possibly inflict on the defender's network).

Figures 4-1 and 4-2 illustrate a problem instance in which $Q = 3$ and $T = 2$. The r-values are stated for each time period next to each node. Consider the case in which no nodes are initially protected, and the attacker influences nodes 1, 2, 6, 8, and 9 at $t = 0$ (Figure 4-1a). As the result of this action, nodes 3 and 5 become influenced at $t = 1$, because nodes 1, 2, and 8 are influenced at $t = 0$ (Figure 4-1b). Nodes 4 and 7 become influenced at $t = 2$ (Figure 4-1c). Hence, the attacker earns a reward of 480. Next, suppose that the defender protects nodes 6 and 9. Then an optimal response

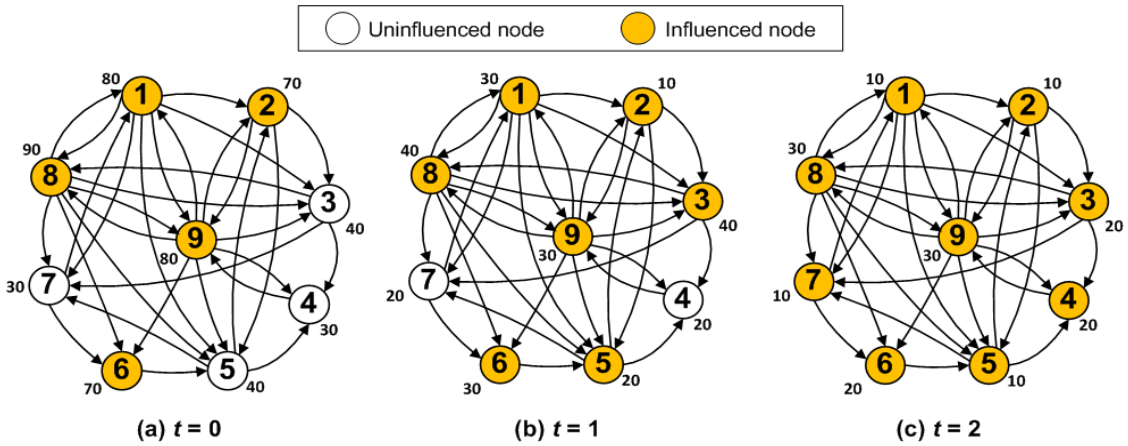

Figure 4-1. An instance with $Q = 3$ and $T = 2$, in the absence of protected nodes.

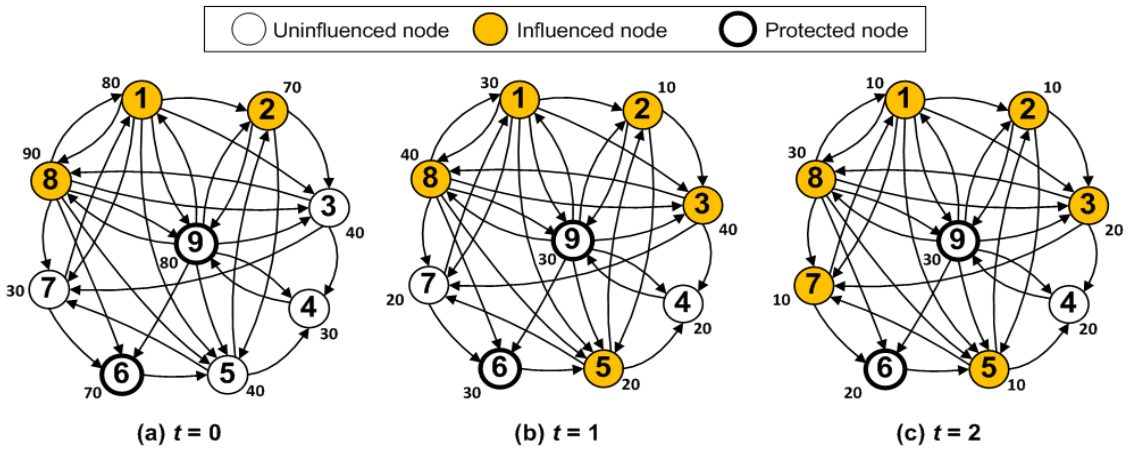

Figure 4-2. An instance with $Q = 3$ and $T = 2$, with nodes 6 and 9 protected by the defender.

from the attacker is to influence nodes 1, 2, and 8 (Figure 4-2a). Although nodes 3 and 5 become influenced at $t = 1$ (Figure 4-2b), only node 7 will be influenced at $t = 2$ (Figure 4-2c). In this case, the attacker's reward reduces to 310.

This problem belongs to the class of Stackelberg leader-follower games [73], because the two players make their actions in turns, where the follower (attacker) operates with full knowledge of the leader's (defender's) decision. A popular approach for solving these problems models them as two-stage *interdiction* problems, which are

then solved via bilevel programming methods. Brown et al. [13, 14] provide summaries on the various applications of interdiction, mostly from a homeland security perspective (see also recent surveys and discussions in [66, 67]).

While we refer to those surveys for a treatment of the history of interdiction development, we note that a common method of approaching the solution of interdiction problems transforms the bilevel min-max problem into a nonlinear minimization problem by dualizing the (attacker's) inner maximization problem, e.g., as done by Wood [75]. However, this approach assumes the existence of a strong dual formulation for the attacker's optimization problem, which (as we will show) is not easily obtainable in this case, because the attacker's problem is NP-hard in the strong sense. Hence, the common methodology used to solve these defender-attacker problems is not applicable to the influence interdiction problem that we consider, which necessitates a new approach that we will explore in this chapter.

The attacker's problem that we consider in this chapter is related to the classical *dominating set* problem [62]. In the dominating set problem, a minimum-cardinality subset of nodes D is sought in an undirected graph $G = (V, E)$ such that every node in $V \backslash D$ is adjacent to at least one of the nodes in D [35]. Several variants of dominating set problem have been studied in the literature, including the *connected dominating set* problem, in which the subset D needs to be a connected graph [12], and the *power dominating set*, in which nodes in D must dominate nodes and arcs in G [39, 77].

In the context of the problems we study in this chapter, "domination" is a special case of influence in which there is a single time period, and a single node can influence all of its adjacent nodes. This concept can be extended to domination via multiple links in a network as well. For instance, Wu et al. [76] study an extended version of the dominating set problem in which a node is influenced either by its dominating neighbor or by some k dominating nodes that can reach the node in two hops. Kempe et al. [42] consider the problem of identifying a set of some k nodes to initially influence, with

the aim of influencing as many nodes as possible (until no more nodes can become influenced, e.g., $T \equiv |V|$). The authors consider two *diffusion models*, which dictate how influence spreads across the network. One is a threshold model similar to the one we describe above, and the other is an *independence cascade model*, where each influenced node has one opportunity to influence a neighbor, and does so in a probabilistic manner. The authors show that their influence function is submodular, which enables them to provide a $(1 - 1/e)$ greedy approximation algorithm (see [52] for approximation algorithm theory as applied to submodular functions).

Leskovec et al. [44] consider a model in which influence spreads to all adjacent nodes, as information spreads throughout a set of networked blogs. The goal is to monitor a set of blogs (nodes) that detect spreading information as quickly as possible. Chen et al. [19] provide improved scalable algorithms for approximating the maximum influence problem addressed in [42]. From a different perspective, the influence problem can also be formulated as finding a minimum cardinality set of nodes, which when initially influenced, will eventually lead to the influence of all nodes in the network. Dinh et al. [28] show that the number of initially influenced nodes is $\Omega(n)$, and provide an $O(1)$-approximation algorithm in power-law networks and $O(\log n)$-approximation algorithm in general networks. Finally, Shen et al. [64] investigate this problem in multiplex networks, inspired by the scenario in which users can simultaneously spread influence into multiple networks.

The preceding works focus on maximization of influence without the presence of node protection. Conversely, instead of maximizing influence, some research has recently been proposed to contain the spread of misinformation. Shen at al. [63] examine a node deletion problem with the aim of minimizing the maximum component size of a graph. Viewing “deleted” nodes as protected nodes, this problem thereby limits the maximum number of nodes that could be influenced from a single source when

$Q = 1$. From a computer network perspective, [55, 70] explore the deployment of benign computer worms to counteract malicious code in an active fashion.

Relevant to our study, Budak et al. [15] address the problem of influence limitation, given a single initial point of influence. As in our study, their problem protects nodes against influence, but protected nodes also inject their own "good" influence into the network (as opposed to the adversary's "bad" influence). The authors assume that if good and bad information simultaneously arrive at a node, the good information will be adopted, and that the set of nodes spreading misinformation is known a priori. Nguyen et al. [54] study the problem of finding a smallest set of nodes from which good influence serves to contain the spread of misinformation. They investigate both the case in which the originating nodes that spread misinformation are known, and the case in which they are unknown.

There are several key differences between the study proposed in this chapter and those that precede it. For one, we seek an optimal solution to the influence interdiction problem in lieu of an approximation scheme. Moreover, the reward function earned by the attacker is more general, and can capture the case in which the attacker's benefit in influencing nodes is discounted as a function of time. As we will describe in the next section, the only assumption made on the reward function is that the attacker's reward for influencing a node is a nonincreasing function of the time at which it is first influenced: This function need be neither concave nor convex. As such, our problem characterization captures different problem classes than those that have been studied in the literature.

Exact optimization algorithms for this problem will naturally require considerably more computational effort than the aforementioned approximation algorithms, especially those that are scalable to large-scale social networks. Of course, exact algorithms are useful in creating benchmarks for heuristic schemes, so that one can empirically (on smaller networks) determine the effectiveness of such algorithms in finding near-optimal

solutions. Still, the interdiction of influence has many applications on smaller scale networks, where exact algorithms may be applied in practical situations. For instance, if $Q = 1$ and $T = 1$, with all r-values equalling 1, then the problem seeks to minimize the maximum number of nodes that could be dominated by a set of unprotected nodes. As T grows, the notion of domination is relaxed. Defense networks may seek to fortify physical positions against attack, where these positions (represented as nodes) are vulnerable if some Q locations decide to simultaneously attack. Note that the spread of influence in this case may refer to advancing military units that capture positions as they attack, using them as forward points for further attacks.

The remainder of this chapter is organized as follows. We formally define the influence interdiction problem in Section 4.2 and provide a two-stage mathematical formulation that models the problem. In Section 4.3, we examine a set of alternative cutting-plane approaches to solving this problem. Section 4.4 revisits the attacker's problem formulated in Section 4.2, exploring formulations that employ fewer binary variables than the "natural" formulation for the problem. Finally, we investigate the efficacy of our algorithms on randomly generated instances in Section 4.5.

4.2 Problem Formulation

For each node $i \in V$, define the set of incoming neighbors of i as $V^-(i) = \{j \in V : (j,i) \in A\}$, and the set of outgoing neighbors of i as $V^+(i) = \{j \in V : (i,j) \in A\}$. Let $\mathcal{T} = \{1, \dots, T\}$ be the set of time periods. Recall that an unprotected node $i \in V$ that is not influenced at time $t-1$ becomes influenced at time $t \in \mathcal{T}$ if at least Q nodes in $V^-(i)$ are influenced at time $t-1$, where we refer to Q as the threshold influence parameter. Also, recall that the attacker earns a reward of r_i^t if node $i \in V$ is influenced at time $t \in \mathcal{T} \cup \{0\}$ for the first time, where $r_i^0 \geq \cdots \geq r_i^T$. There exists a cost of $c_i,\ i \in V$, for the attacker to influence node i at time zero. Similarly, the defender incurs a cost of $b_i,\ i \in V$, to protect node i. In our model, the defender (attacker) has a budget of B (D) to protect (initially influence) nodes.

In order to formulate this problem, we first define two sets of binary decision variables $x_i,\ i \in V$, and y_i^0. In our model, $x_i = 1$ if the defender protects node $i \in V$, and $x_i = 0$ otherwise. Also, $y_i^0 = 1$ if node $i \in V$ is influenced by the attacker at time zero, and $y_i^0 = 0$ otherwise. Additionally, we introduce binary decision variables $y_i^t = 1,\ t \in \mathcal{T}$, if node $i \in V$ is influenced at time t, and $y_i^t = 0$ otherwise. Note that we have separated y^0-variables from y^t-variables to emphasize the difference between influence at $t = 0$ and $t > 0$, because the latter results from the spread of influence. The defender's problem can be formulated as follows.

$$\min \quad z(x) \tag{4–1a}$$

$$\text{s.t.} \quad b^T x \leq B \tag{4–1b}$$

$$x_i \in \{0,1\} \qquad \forall i \in V, \tag{4–1c}$$

where $z(x)$ is the optimal objective value of the attacker's problem, which can be computed by solving the following integer program given some fixed value of $x = \bar{x}$:

$$\textbf{ATT1}(\bar{x}) : z(\bar{x}) = \max \quad \sum_{i \in V} \left(r_i^0 y_i^0 + \sum_{t=1}^{T} r_i^t (y_i^t - y_i^{t-1}) \right) \tag{4–2a}$$

$$\text{s.t.} \quad y_i^t \leq 1 - \bar{x}_i \qquad \forall i \in V,\ t \in \mathcal{T} \cup \{0\} \tag{4–2b}$$

$$Qy_i^t \leq Qy_i^0 + \sum_{j \in V^-(i)} y_j^{t-1} \qquad \forall i \in V,\ t \in \mathcal{T} \tag{4–2c}$$

$$c^T y^0 \leq D \tag{4–2d}$$

$$y_i^0 \in \{0,1\} \qquad \forall i \in V \tag{4–2e}$$

$$y_i^t \in \{0,1\} \qquad \forall i \in V,\ t \in \mathcal{T}. \tag{4–2f}$$

The objective function 4–1a reflects the defender's goal of minimizing the maximum reward earned by the attacker (computed by solving 4–2). The defender's budget limit and the binariness of the x-variables are enforced by Constraints 4–1b and 4–1c, respectively. The objective function 4–2a represents the attacker's reward, noting that

$y_i^0 = 1$ if node $i \in V$ is initially influenced, and $y_i^t - y_i^{t-1} = 1$ if node $i \in V$ is influenced for the first time at time $t \in \mathcal{T}$. Constraint 4–2b implies that a protected node can never be influenced by the attacker. Constraints 4–2c governs the spread of influence: If node $i \in V$ is initially influenced, then the right-hand-side (RHS) of Constraints 4–2c will be at least Q for all $t \in \mathcal{T}$, implying that node i will remain influenced at all time periods. Now, suppose that node $i \in V$ is not initially influenced, and consider Constraint 4–2c for node i and time $t \in \mathcal{T}$. The constraint implies that node i can be influenced at time t if and only if $\sum_{j \in V^-(i)} y_j^{t-1} \geq Q$, i.e., if and only if at least Q nodes adjacent to node i are influenced at time $t-1$. Note that for any node-time pair $i \in V$ and $t \in \mathcal{T}$, y_i^t is present with a nonnegative coefficient in the objective function (due to nonincreasing values for node i's rewards over time). Therefore, there exists an optimal solution in which $y_i^t = 1$ if and only if either node i is initially influenced, or at least Q nodes adjacent to node i are influenced at time $t-1$. Finally, Constraint 4–2d enforces the attacker's budget limit, and Constraints 4–2e and 4–2f restrict the y-variables to be binary-valued.

In the rest of this chapter, we define $X = \{x \in \{0,1\}^{|V|} : b^T x \leq B\}$ as the set of possible actions for the defender. Given $\bar{x} \in X$, we also define $Y(\bar{x}) = \{y^0 \in \{0,1\}^{|V|} : c^T y^0 \leq D,\ y_i^0 \leq 1 - \bar{x}_i,\ \forall i \in V\}$ as the set of available actions for the attacker at time zero when the defender chooses $\bar{x}$.

We may also wish to consider the case in which the influence threshold value depends on the node being influenced, and so node i becomes influenced at time t if some Q_i nodes in $V^-(i)$ are influenced at time $t-1$. This case can be transformed to the case in which all nodes have a common threshold value, Q. To see this, let $Q = \max_{i \in V}\{Q_i\}$. Create a set of Q dummy nodes that are impossible to protect and free for the attacker to initially influence, and let the reward for influencing these nodes equal 0 at all time periods. For each $i \in V$, create an arc from $Q - Q_i$ of the dummy nodes to node i. Because an optimal solution exists in which all of these dummy nodes would be initially influenced, only Q_i more nodes in $V^-(i)$ from the original graph must

be influenced in order to influence node i, as desired. Hence, for simplicity, we use the common threshold value of Q in this chapter. □

We finish this section by observing that the attacker's problem is strongly NP-hard. To see this, we sketch a (polynomial) reduction from the *dominating set* problem [35] to a variant of the attacker's problem. In the dominating set problem, we seek a subset D of nodes in an undirected graph $\bar{G}(\bar{V}, \bar{E})$ such that each node in $\bar{V} \setminus D$ is adjacent to at least one node in D, and such that $|D| \leq \delta$ for some given positive integer δ. Now, consider the attacker's problem with $Q = 1$ and $T = 1$. Let the attacker's problem network $G(\bar{V}, A)$ consist of the same node set as in the dominating set instance, and let A contain two directed arcs, (i, j) and (j, i), for each $(i, j) \in \bar{E}$. Also, define $B = \delta$ and $b_i = r_i^0 = r_i^1 = 1, \ \forall i \in \bar{V}$. There exists a dominating set having δ nodes if and only if there exists a solution to the attacker's problem with reward $|\bar{V}|$. Hence, the attacker's problem is strongly NP-hard. Moreover, the defender's problem is also NP-hard, because evaluating $z(x)$ cannot be done in polynomial time unless P = NP.

4.3 Exact Solution Method

In this section, we provide a cutting-plane scheme to solve the problem considered in this chapter. In Section 4.3.1, we state a reformulation to 4–1 that is amenable to a cutting-plane algorithm, and provide objective function bounds that will be useful in our algorithm. In Section 4.3.2, we develop a set of valid inequalities for the problem, and state our cutting-plane algorithm. Then, in order to improve the efficacy of the proposed algorithm, we devise a stronger class of cutting planes in Section 4.3.3.

4.3.1 Reformulation and Objective Bounds

The inherent difficulty in solving Problem 4–1 is due to the nonconvexity of the attacker's problem, which prohibits us from readily obtaining a strong (minimization) dual to Problem 4–2 and employing standard interdiction models as used in [75]. In order to devise a cutting-plane algorithm, we start by proposing a reformulation of the problem considered in this chapter.

We reformulate 4–1 by introducing a variable z, and minimizing z subject to the restriction that $x \in X$ and $z \geq z(x)$. We refer to a pair $(\bar{x}, \bar{z})$ as a *two-stage feasible solution* if $\bar{x} \in X$ and $\bar{z} \geq z(\bar{x})$. Defining Ω as the set of all two-stage feasible solutions, we obtain the following reformulation for the defender's problem:

$$\textbf{DEF}: \quad \min \ z \tag{4–3a}$$

$$\text{s.t.} \ x \in X \tag{4–3b}$$

$$(x, z) \in \Omega. \tag{4–3c}$$

Note that an optimal solution $(x^\star, z^\star)$ to Problem 4–3 satisfies $z^\star = z(x^\star)$, because Problem 4–3 is a minimization program.

Let $\bar{\Omega} \supseteq \Omega$ be a feasible region induced by a set of affine inequalities, and define DEF-R as the relaxation of Problem 4–3 obtained by replacing Ω with $\bar{\Omega}$. The motivation for introducing DEF-R stems from the fact that exponentially many inequalities may be required for the explicit definition of Ω. Hence, our approach starts with an initial $\bar{\Omega}$ defined by a small (polynomial-size) set of inequalities. If an optimal solution $(\bar{x}, \bar{z})$ to DEF-R is two-stage feasible, then it must be also optimal to 4–1, because DEF-R is a relaxation of Problem 4–1. Otherwise, we can augment DEF-R with a cutting plane (as discussed in Sections 4.3.2 and 4.3.3) and re-solve DEF-R in an iterative fashion until a two-stage feasible solution is found. We start by computing lower and upper bounds for the optimal objective value of the attacker's problem.

Lemma 5. *Let $i_{(j)}$, $1 \leq j \leq |V|$, be the node having the j^{th} largest reward at $t = 0$. Denote by q_1 the largest integer such that $\sum_{i \in V'} c_i \leq D$, $\forall V' \subseteq V : |V'| = q_1$. Also, denote by p_2 the largest integer such that $\sum_{i \in V'} b_i \leq B$ for some $V' \subseteq V : |V'| = p_2$. Define:*

$$z_{\min} = \sum_{j=p_2+1}^{\min\{|V|,\ p_2+q_1\}} r^0_{i_{(j)}}.$$

We have:

$$z(x) \geq z_{\min} \qquad \forall x \in X. \tag{4–4}$$

Proof. Let $M = \min\{|V|,\ p_2 + q_1\}$, and define $J = \{1, \dots, M\}$. Observe that p_2 is the maximum number of nodes that can be protected by the defender, and that the attacker can influence any set of q_1 nodes at time 0. Thus, there must exist a subset $\bar{J} \subseteq J,\ |\bar{J}| = \min\{|V| - p_2,\ q_1\}$, that the attacker can initially influence. The attacker can thus always achieve an initial reward given by the sum of the M smallest r^0-values in J, which equals $z_{\min}$. This completes the proof. □

Lemma 6. *Let q_2 be the largest integer such that $\sum_{i \in V'} c_i \leq D$ for some $V' \subseteq V$: $|V'| = q_2$. Given a defender's decision vector $\bar{x}$, an upper bound on the attacker's optimal objective value is obtained by solving the following problem.*

$$z_{\max}(\bar{x}) = \max \quad \sum_{i \in V} (1 - \bar{x}_i) \left(r_i^0 y_i^0 + r_i^1 (1 - y_i^0) \right) \tag{4–5a}$$

$$\text{s.t.} \quad \sum_{i \in V} y_i^0 \leq q_2 \tag{4–5b}$$

$$0 \leq y_i^0 \leq 1 \qquad \forall i \in V \tag{4–5c}$$

Proof. If the attacker adopts an initial attack, $\bar{y}^0 \in Y(\bar{x})$, then the attacker's reward from node $i \in V$ is 0 if node i was protected, r_i^0 if node i was not protected and $\bar{y}_i^0 = 1$, and is no more than r_i^1 if node i was not protected and $\bar{y}_i^0 = 0$. The latter bound is valid because if an unprotected node $i \in V$ is not initially influenced, then it cannot be influenced earlier than time 1, and $r_i^1 \geq r_i^t,\ \forall t = 2, \dots, T$. Therefore

$$z(\bar{x}) \leq \max \quad \sum_{i \in V} (1 - \bar{x}_i) \left(r_i^0 \bar{y}_i^0 + r_i^1 (1 - \bar{y}_i^0) \right),$$

over all $\bar{y}^0 \in Y(\bar{x})$. Because the feasible region defined by Constraints 4–5b–4–5c contains $Y(\bar{x})$, we have that $z(\bar{x}) \leq z_{\max}(\bar{x})$, and this completes the proof. □

Note that Problem 4–5 can be optimized in $\mathcal{O}(|V|\log(|V|))$ steps by sorting the $(1-\bar{x}_i)(r_i^0 - r_i^1)$-values in nonincreasing order, and setting $y_i^0 = 1$ for each node $i \in V$ corresponding to the q_2-largest such coefficients.

The bound $z(\bar{x}) \leq z_{\max}(\bar{x})$ is valid for any $\bar{x} \in X$, and so one strategy may enumerate several candidate solutions $\bar{x} \in X$, compute $z_{\max}(\bar{x})$ for each vector, and obtain the minimum such value as a valid upper bound. Additionally, we can solve the following optimization problem.

$$z_{\max} = \min \quad z_{\max}(x) \tag{4–7a}$$

$$\text{s.t.} \quad x \in X. \tag{4–7b}$$

Observe that Problem 4–7 is a two-stage program in which the feasible region of the inner problem is independent of x-variables. This allows us to state Problem 4–7 as a linear mixed-integer program by dualizing the inner problem. Let α and $\beta_i,\ i \in V$, be the dual variables corresponding to Constraints 4–5b and 4–5c, respectively. We obtain the following reformulation of 4–7.

$$z_{\max} = \sum_{i \in V} r_i^1 + \min \quad q_2\alpha + \sum_{i \in V} \left(\beta_i - r_i^1 x_i\right) \tag{4–8a}$$

$$\text{s.t.} \quad \alpha + \beta_i + (r_i^0 - r_i^1)x_i \geq (r_i^0 - r_i^1) \qquad \forall i \in V \tag{4–8b}$$

$$\alpha \geq 0 \tag{4–8c}$$

$$\beta_i \geq 0 \qquad \forall i \in V \tag{4–8d}$$

$$x \in X. \tag{4–8e}$$

Note that Problem 4–8 has to be solved only once in order to obtain an upper bound for the problem, and hence will not likely represent a substantial portion of the time required to solve the overall model we investigate here.

An alternative strategy is to heuristically select some $\bar{x} \in X$, e.g., by using the following greedy algorithm. Initialize $\bar{x}_i = 0,\ \forall i \in V$, and set a "remaining budget" value

$\bar{B} = B$. Find an index i such that r_i^0 is maximized over all $i \in V$ such that $\bar{x}_i = 0$ and b_i is not more than $\bar{B}$. If no such index i exists, then set the upper bound to $z_{\max}(\bar{x})$. Else, set $\bar{x}_i = 1$, reduce $\bar{B}$ by b_i, and reiterate. In our initial computational experiments, we compare the effectiveness of the exact formulation 4–8 versus the use of this greedy algorithm, and employ the most effective one in our computational study. □

4.3.2 Cutting-Plane Algorithm

In this section, we provide valid inequalities for DEF-R, and we propose a cutting-plane scheme for identifying an optimal solution to 4–1.

Consider $\bar{x} \in X$ and suppose that $\bar{y} = (\bar{y}^0, \dots, \bar{y}^T)$ is optimal to ATT1($\bar{x}$). We define $\bar{\tau}_i,\ i \in V$, as the earliest time that node i is influenced in the solution $\bar{y}$. We use the convention $\bar{\tau}_i = T + 1$ if node $i \in V$ is never influenced by the attacker, and we let $r_i^{T+1} = 0$. Consider the vector $\bar{\tau} = (\bar{\tau}_1, \dots, \bar{\tau}_{|V|})$, and for all $i \in V$, define $R_i^{\bar{\tau}}$ as the set of all unprotected nodes $j \in V$ such that there exists a directed path from node i to node j using $\bar{\tau}_j$ or fewer arcs in A (and hence, $i \in R_i^{\bar{\tau}}$).

Lemma 7. *Consider a solution $\bar{x} \in X$ in which $\bar{x}_i = 0$ for some node $i \in V$. Let $\bar{y} = (\bar{y}^0, \dots, \bar{y}^T)$ be an optimal solution to ATT1($\bar{x}$), with corresponding vector $\bar{\tau}$. Suppose that the solution $\hat{x}$, which is identical to $\bar{x}$ with the exception of setting $\hat{x}_i = 1$, is feasible to 4–1. Then we have:*

$$z(\hat{x}) \geq z(\bar{x}) - \sum_{j \in R_i^{\bar{\tau}}} r_j^{\bar{\tau}_j}. \tag{4–9}$$

Proof. We start by constructing a solution $\hat{y} = (\hat{y}^0, \dots, \hat{y}^T)$ to ATT1($\hat{x}$) as follows. Let $\hat{y}_j^t = 0,\ \forall j \in R_i^{\bar{\tau}},\ t \in \mathcal{T} \cup \{0\}$, and $\hat{y}_j^t = \bar{y}_j^t,\ \forall j \in V \backslash R_i^{\bar{\tau}},\ t \in \mathcal{T} \cup \{0\}$. We first prove that $\hat{y}$ is feasible to ATT1($\hat{x}$).

Note that $\hat{y}_j^t \leq \bar{y}_j^t,\ \forall j \in V,\ t \in \mathcal{T} \cup \{0\}$, and in particular, $\hat{y}_i^t = 0$ (because $i \in R_i^{\bar{\tau}}$). Hence, $\hat{y}$ does not violate Constraints 4–2b and 4–2d. Moreover, all $\hat{y}$-values remain binary to satisfy 4–2e and 4–2f. Constraints 4–2c corresponding to node $j \in V$, with $\hat{y}_j^0 = 1$ or $\hat{y}_j^T = 0$ are clearly satisfied. For Constraints 4–2c corresponding to node $j \in V$ for which $\hat{y}_j^0 = 0$ and $\hat{y}_j^T = 1$, observe that $\hat{y}_j^t = \bar{y}_j^t = 0,\ \forall t = 0, \dots, \bar{\tau}_j - 1$, and

$\hat{y}_j^t = \bar{y}_j^t = 1,\ \forall t = \bar{\tau}_j, \dots, T$. It is sufficient to show that $\hat{y}_j^{\bar{\tau}_j} = 1$ satisfies Constraint 4–2c for node j and time $\bar{\tau}_j$. Note that this constraint was satisfied in the solution $\bar{y}$, and hence, if $\hat{y}_k^{\bar{\tau}_j - 1} = \bar{y}_k^{\bar{\tau}_j - 1},\ \forall k \in V^-(j)$, then the result holds. Recall that the length of any shortest path from node i to node j in G exceeds $\bar{\tau}_j$ (or else, $j \in R_i^{\bar{\tau}}$ implying $\hat{y}_j^t = 0,\ \forall t \in \mathcal{T} \cup \{0\}$). Consider any $k \in V^-(j)$ such that $\bar{y}_k^{\bar{\tau}_j - 1} = 1$. The shortest-path length from node i to node k must exceed $\bar{\tau}_j - 1$ (or else, the shortest-path length from node i to node j would not exceed $\bar{\tau}_j$). Because $\bar{y}_k^{\bar{\tau}_j - 1} = 1$, we have that $\bar{\tau}_k \leq \bar{\tau}_j - 1$, and because $\bar{\tau}_j - 1 < (\text{shrotest-path length from node } i \text{ to node } k)$, we have that $k \in V \backslash R_i^{\bar{\tau}}$. This implies that $\hat{y}_k^t = 1,\ \forall t = \bar{\tau}_k, \dots, T$, and in particular, $\hat{y}_k^{\bar{\tau}_j - 1} = 1$, i.e., $\hat{y}_k^{\bar{\tau}_j - 1} = 1$ if $\bar{y}_k^{\bar{\tau}_j - 1} = 1,\ \forall k \in V^-(j)$. Therefore, $\hat{y}$ remains feasible.

Second, the attacker's objective, $\hat{z}$, of this solution is given by:

$$\begin{aligned}
\hat{z} &= \sum_{j \in V} r_j^0 \hat{y}_j^0 + \sum_{j \in V} \sum_{t \in \mathcal{T}} r_j^t (\hat{y}_j^t - \hat{y}_j^{t+1}) \\
&= \sum_{j \in V \backslash R_i^{\bar{\tau}}} r_j^0 \bar{y}_j^0 + \sum_{j \in V \backslash R_i^{\bar{\tau}}} \sum_{t \in \mathcal{T}} r_j^t (\bar{y}_j^t - \bar{y}_j^{t+1}) \\
&= z(\bar{x}) - \left(\sum_{j \in R_i^{\bar{\tau}}} r_j^0 \bar{y}_j^0 + \sum_{j \in R_i^{\bar{\tau}}} \sum_{t \in \mathcal{T}} r_j^t (\bar{y}_j^t - \bar{y}_j^{t+1}) \right) \\
&= z(\bar{x}) - \sum_{j \in R_i^{\bar{\tau}}} r_j^{\bar{\tau}_j}.
\end{aligned}$$

Because $z(\hat{x}) \geq \hat{z}$, the lemma holds. □

For any given $\bar{x} \in X$, define $P_{\bar{x}}$ as the set of protected nodes in $\bar{x}$. Furthermore, let $\bar{y} = (\bar{y}^0, \dots, \bar{y}^T)$ be an optimal solution to ATT1($\bar{x}$), and define $V_{\bar{x}}$ as the set of all influenced nodes in solution $\bar{y}$. We introduce our first valid inequality for 4–3 in the next theorem.

Theorem 4.1. *Let $(\bar{x}, \tilde{z})$ be an optimal solution to DEF-R, and suppose that $\tilde{z} < z(\bar{x})$. Let $\bar{y}$ be an optimal solution to ATT1($\bar{x}$), with corresponding vector $\bar{\tau}$. The following*

inequality:

$$z \geq z(\bar{x}) - \sum_{i \in V_{\bar{x}}} \left(\min \left\{ z(\bar{x}) - z_{\min},\ \sum_{j \in R_i^{\bar{\tau}}} r_j^{\bar{\tau}_j} \right\} \right) x_i, \tag{4–10}$$

is valid to 4–3 and cuts off $(\bar{x}, \tilde{z})$.

Proof. First, consider any solution $\hat{x} \in X$ such that $\hat{x}_i = 0,\ \forall i \in V_{\bar{x}}$. The attacker's solution $\bar{y}$ is still feasible to ATT1($\hat{x}$). Hence, $z(\hat{x}) \geq z(\bar{x})$ in this case, and so 4–10 is valid. In particular, setting $\hat{x} = \bar{x}$ satisfies the condition $\hat{x}_i = 0,\ \forall i \in V_{\bar{x}}$, requiring that $z \geq z(\bar{x})$ at this point. Hence, 4–10 cuts off $(\bar{x}, \tilde{z})$ by the assumption that $\tilde{z} < z(\bar{x})$.

Next, consider any solution $\hat{x} \in X$ such that $P_{\hat{x}} \cap V_{\bar{x}} \neq \emptyset$, and define $M_i = \sum_{j \in R_i^{\bar{\tau}}} r_j^{\bar{\tau}_j},\ \forall i \in P_{\hat{x}} \cap V_{\bar{x}}$. If $M_i \geq z(\bar{x}) - z_{\min}$ for some $i \in P_{\hat{x}} \cap V_{\bar{x}}$, then the RHS of 4–10 is no more than $z_{\min}$, and 4–10 is valid by Lemma 5. Else, suppose that $M_i < z(\bar{x}) - z_{\min},\ \forall i \in P_{\hat{x}} \cap V_{\bar{x}}$. In this case, define $R^{\bar{\tau}}_{P_{\hat{x}} \cap V_{\bar{x}}}$, as the set of all nodes $j \in V_{\bar{x}}$ such that the shortest path in G from any node $i \in P_{\hat{x}} \cap V_{\bar{x}}$ to node j is no more than $\bar{\tau}_j$. An identical proof to Lemma 2 shows that

$$z \geq z(\bar{x}) - \sum_{j \in R^{\bar{\tau}}_{P_{\hat{x}} \cap V_{\bar{x}}}} r_j^{\bar{\tau}_j},$$

is valid. Because

$$\sum_{i \in P_{\hat{x}} \cap V_{\bar{x}}} \sum_{j \in R_i^{\bar{\tau}}} r_j^{\bar{\tau}_j} \geq \sum_{j \in R^{\bar{\tau}}_{P_{\hat{x}} \cap V_{\bar{x}}}} r_j^{\bar{\tau}_j},$$

it follows that

$$z \geq z(\bar{x}) - \sum_{i \in P_{\hat{x}} \cap V_{\bar{x}}} \sum_{j \in R_i^{\bar{\tau}}} r_j^{\bar{\tau}_j},$$

which establishes the validity of 4–10. □

The following cutting-plane algorithm, which we call CPA, is then given as follows.

Step 0 (Initialization). Set the upper bound $UB = z_{\max}$ and the lower bound $LB = z_{\min}$. Let $\bar{\Omega} = \{(x, z) : x \in X,\ z \geq z_{\min}\}$.

Step 1 (Lower Bound). Identify an optimal solution $(\bar{x}, \tilde{z})$ to DEF-R. Set $LB = \tilde{z}$, and proceed to Step 2.

Step 2 (Upper Bound). Solve ATT1($\bar{x}$). If $z(\bar{x}) < UB$, then set $UB = z(\bar{x})$, and let $\bar{x}$ be the incumbent solution to Problem 4–1. In either case, proceed to Step 3.

Step 3 (Termination/Cut Routine). If $LB = UB$, then terminate with the incumbent solution being optimal. Otherwise, add 4–10 to $\bar{\Omega}$, and return to Step 1.

At each iteration, CPA either terminates with an optimal solution in Step 3, or the identified solution $(\bar{x}, \tilde{z})$ in Step 1 will be cut off by the inequality added in Step 3. Note that X is a finite set, and that if some solution x' is encountered as the optimal solution in Step 1 in two different iterations k_1 and k_2 of CPA, then $LB = UB$ after iteration k_2. This behavior is due to the fact that inequality 4–10 was added for $x = x'$ after iteration k_1; hence, at iteration k_2, this inequality forces $z \geq z(x')$. As a result, CPA converges to an optimal solution in a finite number of steps.

4.3.3 Spread Network Inequalities

In Section 4.3.3.1, we explore the development of alternative valid inequalities that are at least as strong as 4–10, with the aim of reducing the number of iterations required by CPA. In Section 4.3.3.2, we examine a method for further strengthening these inequalities.

4.3.3.1 Spread-network-based cutting planes

For $\bar{x} \in X$, consider an optimal solution $\bar{y} = (\bar{y}^0, \ldots, \bar{y}^T)$, and its associated vector $\bar{\tau}$. Define $V^t_{\bar{x}} = \{i \in V : \bar{\tau}_i = t\},\ t \in \mathcal{T} \cup \{0\}$, and observe that $V_{\bar{x}} = \bigcup_{t=0}^{T} V^t_{\bar{x}}$. We denote by $G_{\bar{x}}(V_{\bar{x}}, A_{\bar{x}})$ an acyclic time-expanded network, called the *spread network*, with $T + 1$ time levels. For node $i \in V^t_{\bar{x}}$ at time $t \in \mathcal{T}$, there exist at least Q nodes h such that $(h, i) \in A$ and $\bar{\tau}_h \leq t - 1$. Let $S',\ |S'| = Q$, be any subset of such nodes. Then, for each $i \in V^t_{\bar{x}},\ t \in \mathcal{T}$, add Q arcs $(h, i),\ \forall h \in S'$, to the spread network. Note that our construction implies that the spread network for $\bar{x}$ is not necessarily unique, because more than one such set S' may exist (see Figure 4-3 for an example).

For all $i \in V_{\bar{x}}$, define:

$$O_i = \{h \in V_{\bar{x}} : \text{ there exists a directed path on } G_{\bar{x}} \text{ from node } h \text{ to node } i\},$$

and hence, $i \in O_i$. Next, consider any $\hat{x} \in X$ and define:

$$I_{\hat{x}} = \{i \in V_{\bar{x}} : O_i \cap P_{\hat{x}} \neq \emptyset\},$$

i.e., $I_{\hat{x}}$ is the set of all nodes $i \in V_{\bar{x}}$ that are either protected in $\hat{x}$, or such that there exists a directed path on $G_{\bar{x}}$ from some node $h \in P_{\hat{x}} \cap V_{\bar{x}}$ to node i. In the following lemma and theorem, we derive alternative valid inequalities for 4–3 by using the idea of the spread network.

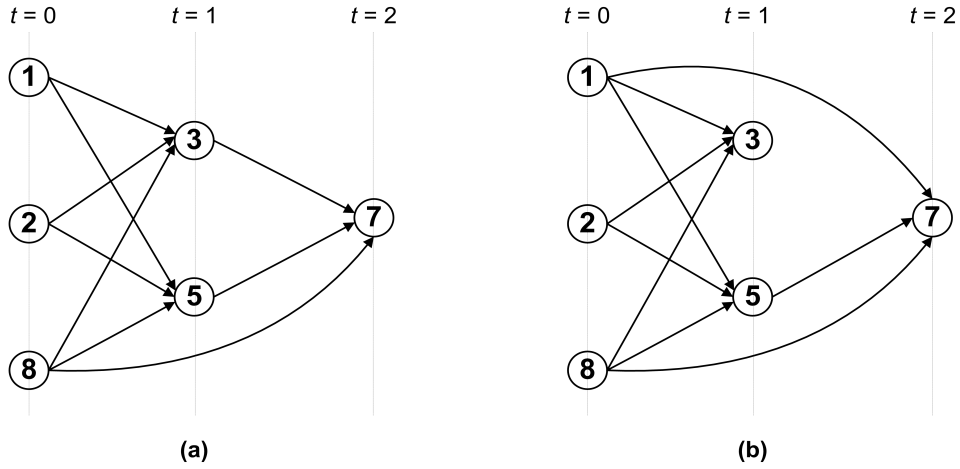

Figure 4-3. Two possible spread networks for Figure 4-2.

Lemma 8. *Given $\bar{x} \in X$, let $(\bar{y}^0, \ldots, \bar{y}^T)$ be optimal to ATT1($\bar{x}$) with corresponding vector $\bar{\tau}$. For any $\hat{x} \in X$, we have:*

$$z(\hat{x}) \geq \sum_{j \in V_{\bar{x}} \setminus I_{\hat{x}}} r_j^{\bar{\tau}_j}. \tag{4–11}$$

Proof. Suppose that the attacker chooses an initial attack, $\hat{y}^0$, by letting $\hat{y}_i^0 = 1$ if $\bar{y}_i^0 = 1$ and $\hat{x}_i = 0$, and $\hat{y}_i^0 = 0$ otherwise, for all $i \in V$. We prove this lemma by showing that if $j \in V_{\bar{x}}^t \setminus I_{\hat{x}}$, then node j will still be influenced at time t when the defender chooses $\hat{x}$. First, note that node j is initially influenced in solution $\hat{y}$ for any node $j \in V_{\bar{x}}^0 \setminus I_{\hat{x}}$. By induction, suppose that for some $t \in \{0, \ldots, T-1\}$, all nodes in $\{\bigcup_{t'=1}^{t} V_{\bar{x}}^{t'}\} \setminus I_{\hat{x}}$ are influenced at time t', when the defender chooses $\hat{x}$ and the attacker chooses $\hat{y}^0$.

Consider any node $j \in V_{\bar{x}}^{t+1} \backslash I_{\hat{x}}$. There exists no directed path from any node in $I_{\hat{x}}$ to node j, or else j would belong to $I_{\hat{x}}$ as well. Thus, there also exists no directed path from any node in $I_{\hat{x}}$ to any node i such that $(i, j) \in A_{\bar{x}}$. For each of the Q nodes $i \in \bigcup_{t'=1}^{t} V_{\bar{x}}^{t'}$ such that $(i, j) \in A_{\bar{x}}$, we have by induction that node i is influenced at time t or earlier in the solution given by $\hat{y}^0$. This implies that node j would be influenced at time $t+1$ given $\hat{x}$ and $\hat{y}^0$, as desired. Therefore, the RHS of 4–11 establishes a lower bound for the attacker's optimal objective value when the defender chooses $\hat{x}$. □

Theorem 4.2. *Let $(\bar{x}, \tilde{z})$ be an optimal solution to DEF-R, and suppose that $z(\bar{x}) > \tilde{z}$. Also, let $\mathcal{G}$ be any (undirected) acyclic graph that is constructed over $V_{\bar{x}}$, and denote by $\mathcal{A}$ its set of arcs. Finally, define $\mathcal{A}_j$, $\forall j \in V_{\bar{x}}$, as the set of arcs $(u, v) \in \mathcal{A}$ such that $u \in O_j$ and $v \in O_j$. Then, the following inequality:*

$$z \geq \sum_{j \in V_{\bar{x}}} r_j^{\bar{\tau}_j} \left(1 - \sum_{i \in O_j} x_i + \sum_{(u,v) \in \mathcal{A}_j} x_u x_v \right), \tag{4–12}$$

is valid to 4–3 and cuts off $(\bar{x}, \tilde{z})$.

Proof. First, note that inequalities 4–12 for $x = \bar{x}$ reduce to $z \geq \sum_{j \in V_{\bar{x}}} r_j^{\bar{\tau}_j} = z(\bar{x})$. Hence, inequalities 4–12 cut off $(\bar{x}, \tilde{z})$ by the assumption that $\tilde{z} < z(\bar{x})$.

Next, consider $(\hat{x}, \hat{z}) \in \Omega$. Define $\rho_j = 1 - \sum_{i \in O_j} \hat{x}_i + \sum_{(u,v) \in \mathcal{A}_j} \hat{x}_u \hat{x}_v$, $\forall j \in V_{\bar{x}}$. Consider the following two cases:

- If $\sum_{i \in O_j} \hat{x}_i = 0$ for some node $j \in V_{\bar{x}}$, then we must have $\sum_{(u,v) \in \mathcal{A}_j} \hat{x}_u \hat{x}_v = 0$, implying that $\rho_j = 1$. Note that our construction of O_j implies that $\sum_{i \in O_j} \hat{x}_i = 0$ if and only if $j \in V_{\bar{x}} \backslash I_{\hat{x}}$.
- If $\sum_{i \in O_j} \hat{x}_i = k > 0$ for some node $j \in V_{\bar{x}}$, then $\sum_{(u,v) \in \mathcal{A}_j} \hat{x}_u \hat{x}_v$ can be at most $k - 1$. Otherwise, there would exist at least k arcs $(u, v) \in \mathcal{A}_j$ that are defined over k nodes in $\mathcal{G}$, which contradicts the acyclic property for $\mathcal{G}$. Therefore, we obtain $\rho_j \leq 0$ in this case.

Given these two cases, we obtain:

$$\sum_{j \in V_{\bar{x}}} \rho_j r_j^{\bar{\tau}_j} = \sum_{j \in V_{\bar{x}} \backslash I_{\hat{x}}} \rho_j r_j^{\bar{\tau}_j} + \sum_{j \in I_{\hat{x}}} \rho_j r_j^{\bar{\tau}_j} \leq \sum_{j \in V_{\bar{x}} \backslash I_{\hat{x}}} r_j^{\bar{\tau}_j} \leq z(\hat{x}),$$

where the last inequality is valid by Lemma 8. This completes the proof. □

Corollary 4. *Let* $(\bar{x}, \tilde{z})$ *be an optimal solution to DEF-R, and suppose that* $z(\bar{x}) > \tilde{z}$. *If* $\mathcal{A} = \emptyset$ *in Theorem 4.2, then the following inequality:*

$$z \geq z(\bar{x}) - \sum_{i \in V_{\bar{x}}} \left(\min \left\{ z(\bar{x}) - z_{\min}, \sum_{j \in V_{\bar{x}}:\ i \in O_j} r_j^{\bar{\tau}_j} \right\} \right) x_i, \tag{4–13}$$

is valid to 4–3 and cuts off $(\bar{x}, \bar{z})$.

Proof. Observe that valid inequalities 4–12 reduce to the following inequality when $\mathcal{A} = \emptyset$:

$$z \geq \sum_{j \in V_{\bar{x}}} r_j^{\bar{\tau}_j} - \sum_{j \in V_{\bar{x}}} r_j^{\bar{\tau}_j} \sum_{i \in O_j} x_i,$$

or equivalently,

$$z \geq z(\bar{x}) - \sum_{i \in V_{\bar{x}}} M_i' x_i, \tag{4–14}$$

where $M_i' = \sum_{j \in V_{\bar{x}}:\ i \in O_j} r_j^{\bar{\tau}_j}$. Hence, 4–14 is a valid inequality that cuts off $(\bar{x}, \bar{z})$. Furthermore, because $M_i' \geq 0,\ \forall i \in V_{\bar{x}}$; $z(x) \geq z_{\min},\ \forall x \in X$; and $x_i \in \{0, 1\},\ \forall i \in V_{\bar{x}}$; 4–14 can be strengthened by replacing M_i' with $\min\{z(\bar{x}) - z_{\min},\ M_i'\}$ using a similar argument given in Theorem 4.1. This modification leads to 4–13 and completes the proof. □

Using Theorem 4.2 and Corollary 4, we can employ CPA equipped with valid inequalities of the form 4–12 or 4–13 instead of 4–10 in Step 3 of CPA. The motivation for using these inequalities stems from the following theorem that compares 4–10 to 4–13.

Theorem 4.3. *Inequality 4–13 is at least as strong as 4–10.*

Proof. Define $\kappa_i = \min\{z(\bar{x}) - z_{\min},\ \sum_{j \in V_{\bar{x}}:\ i \in O_j} r_j^{\bar{\tau}_j}\}$ and $\gamma_i = \min\{z(\bar{x}) - z_{\min},\ \sum_{j \in R_i^{\bar{\tau}}} r_j^{\bar{\tau}_j}\}$ for $i \in V_{\bar{x}}$. We prove the claim by showing

$$\sum_{j \in V_{\bar{x}}:\ i \in O_j} r_j^{\bar{\tau}_j} \leq \sum_{j \in R_i^{\bar{\tau}}} r_j^{\bar{\tau}_j},\ \forall i \in V_{\bar{x}}, \tag{4–15}$$

which implies $\kappa_i \leq \gamma_i,\ \forall i \in V_{\bar{x}}$. Define $O'_i = \{j \in V_{\bar{x}} : i \in O_j\}, \forall i \in V_{\bar{x}}$. It suffices to prove that $O'_i \subseteq R_i^{\bar{\tau}}$ for $i \in V_{\bar{x}}$. If $j \in O'_i$, then there exists a directed path on $G_{\bar{x}}$ from node i to node j. Because node j belongs to $V_{\bar{x}}^{\bar{\tau}_j}$, and $A_{\bar{x}} \subseteq A$, there must exist a directed path on G from node i to node j that consists of $\bar{\tau}_j$ or fewer arcs. Hence $j \in R_i^{\bar{\tau}}$. This completes the proof. □

A similar theorem cannot be stated that compares the strength of inequalities 4–12 and 4–13. If inequality 4–13 was weakened by replacing the coefficients of x_i with $\sum_{j \in V_{\bar{x}}:\ i \in O_j} r_j^{\bar{\tau}_j}$ (instead of the minimum of that term and $z(\bar{x}) - z_{\min}$), then 4–12 would be at least as strong as 4–13 due to the subtraction of quadratic terms present in 4–12.

A further consideration in implementing CPA with valid inequalities 4–12 regards the linearization of the quadratic terms in these inequalities. By restricting the set of arcs that can belong to the set $\mathcal{A}$, over all generated inequalities 4–12, we can limit the number of quadratic terms that must be linearized. We linearize each quadratic term $x_i x_j$ by substituting it with a continuous variable $x_{ij}^L \geq 0$, and including the inequality $x_{ij}^L \geq x_i + x_j - 1$ in DEF-R. (The inequalities $x_{ij}^L \leq x_i$ and $x_{ij}^L \leq x_j$ usually required to linearize this quadratic term are not necessary, because optimization forces each x_{ij}^L-variable to take its smallest value allowed by x_i and x_j.)

4.3.3.2 Spread network modification strategy

Recall that multiple spread networks can be derived for a given $\bar{x} \in X$ and its optimal response $\bar{y} = (\bar{y}^0, \ldots, \bar{y}^T)$, each of which might correspond to a different valid inequality of the form 4–12 or 4–13. Given a candidate spread network, $G_{\bar{x}}$, corresponding to $\bar{x}$ and $\bar{y}$, we seek a mechanism for modifying $G_{\bar{x}}$ to an alternative spread network, $G'_{\bar{x}}$, such that the inequality 4–13 generated corresponding to $G'_{\bar{x}}$ is at least as strong as the one corresponding to $G_{\bar{x}}$.

Theorem 4.4. *Consider a spread network $G_{\bar{x}}(V_{\bar{x}}, A_{\bar{x}})$ for which there exist nodes $i, j, k \in V_{\bar{x}}$ such that $i \in V^-(k)$, $(i, k) \notin A_{\bar{x}}$, $(j, k) \in A_{\bar{x}}$, and a path exists from i to j on $G_{\bar{x}}$. Let $\widetilde{G}_{\bar{x}}$ be a modified spread network obtained by replacing arc (j, k) in $G_{\bar{x}}$ with arc*

(i, k). *The valid inequality 4–13 induced by* $\widetilde{G}_{\bar{x}}$ *is at least as strong as that induced by* $G_{\bar{x}}$.

Proof. Consider the spread networks $G_{\bar{x}}$ and $\widetilde{G}_{\bar{x}}$ written with respect to $\bar{x} \in X$, as defined in the theorem. For each $h \in V_{\bar{x}}$ we again define $O'_h = \{j \in V_{\bar{x}} : h \in O_j\}$ as in Theorem 4.3, with respect to spread network $G_{\bar{x}}$, and $\widetilde{O}'_h$ analogously for $\widetilde{G}_{\bar{x}}$. We prove that $\widetilde{O}'_h \subseteq O'_h$ for all $h \in V_{\bar{x}}$. As a result, the x_h-coefficient for 4–13 generated according to $G_{\bar{x}}$ is at least as large as the corresponding coefficient in 4–13 according to $\widetilde{G}_{\bar{x}}$, which is sufficient to prove the theorem.

Note that $k \in O'_i$ due to the assumption that there exists a path from i to j in $G_{\bar{x}}$, and that arc $(j, k) \in A_{\bar{x}}$. Therefore, the addition of arc (i, k) to $A_{\bar{x}}$ does not change O'_i, and by extension does not affect any other set O'_h, $\forall h \in V_{\bar{x}}$. Next, consider the deletion of arc (j, k) from $A_{\bar{x}}$, which then yields $\widetilde{G}_{\bar{x}}$ and its corresponding $\widetilde{O}'_i$ sets. This arc deletion can only decrease membership within the O'-sets, and so $\widetilde{O}'_h \subseteq O'_h$, for all $h \in V_{\bar{x}}$. This completes the proof. □

Figure 4-4 illustrates an instance of the problem with $T = 2$ and $Q = 3$ in which all rewards equal 1 and $z_{\min} = 1$. For a given $\bar{x}$, let $G_{\bar{x}}$ be a spread network presented in Figure 4-4a. Note that $z(\bar{x}) = 7$. Then, the following inequality

$$z \geq 7 - 3x_1 - 4x_2 - 4x_3 - 3x_4 - 2x_5 - 2x_6 - x_7,$$

is induced by $G_{\bar{x}}$ from Corollary 4. Next, suppose that there exists an arc $(4, 7) \in A$ and note that $(4, 6) \in A_{\bar{x}}$. Using Theorem 4.4, we obtain a modified spread network $\widetilde{G}_{\bar{x}}$ from $G_{\bar{x}}$ by adding arc $(4, 7)$ and removing arc $(6, 7)$. (See Figure 4-4b.) By using Corollary 4 for $\widetilde{G}_{\bar{x}}$, we obtain the inequality

$$z \geq 7 - 3x_1 - 4x_2 - 4x_3 - 3x_4 - 2x_5 - x_6 - x_7,$$

which is stronger than the inequality induced by $G_{\bar{x}}$ due to the x_6-coefficients in these inequalities.

It is worth noting that the inequality 4–12 induced by $\widetilde{G}_{\bar{x}}$ may not necessarily be as strong as that induced by $G_{\bar{x}}$ in general. Let $\mathcal{A} = \{(2,6),(3,6)\}$ be the set of arcs used to generate the quadratic terms in 4–12. Inequalities

$$z \geq 7 - 3x_1 - 4x_2 - 4x_3 - 3x_4 - 2x_5 - 2x_6 - x_7 + 2x_2x_6 + 2x_3x_6, \tag{4–16}$$

and

$$z \geq 7 - 3x_1 - 4x_2 - 4x_3 - 3x_4 - 2x_5 - x_6 - x_7 + x_2x_6 + x_3x_6, \tag{4–17}$$

are induced by $G_{\bar{x}}$ and $\widetilde{G}_{\bar{x}}$, respectively, from Theorem 4.2. To see that 4–16 and 4–17 do not dominate one another, we show that the RHS for one constraint need not always be larger than the RHS for the other constraint. For $x' = (0,0,0,0,0,1,0)$, note that the RHS of inequality 4–16 is 5, while the RHS for 4–17 is 6. However, for $x'' = (0,1,1,0,0,1,0)$, the RHS of 4–16 is 1, and the RHS of 4–17 is 0. □

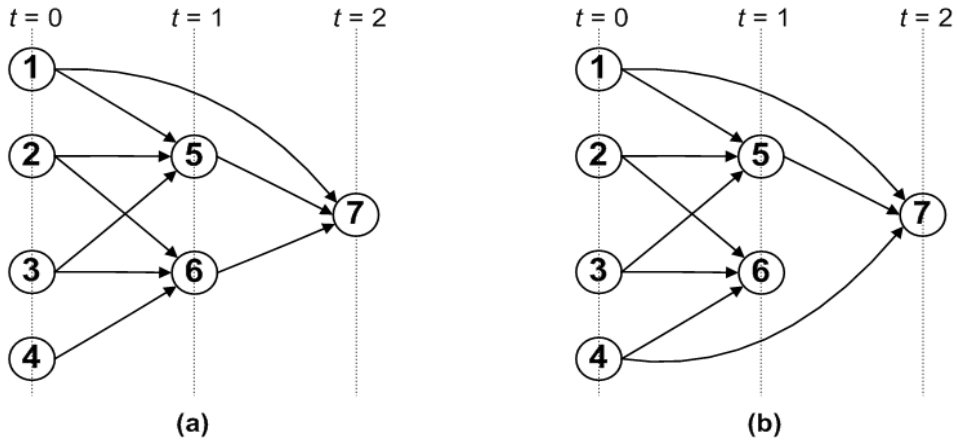

Figure 4-4. Spread network modification using Theorem 4.4.

Algorithm 2 describes our method for modifying a given spread network using the idea of Theorem 4.4 in order to strengthen valid inequality 4–13.

Algorithm 2 examines all candidates for node k (as defined in Theorem 4.4) from among the nodes in $V_{\bar{x}}^T, \ldots, V_{\bar{x}}^2$, in that order. Given a choice of k, the algorithm starts by creating a list L_k, containing all nodes $j \in V^-(k)$ that are influenced at some time

Algorithm 2 Revising an existing spread network using Theorem 4.4

Let $G_{\bar{x}} = (V_{\bar{x}}, A_{\bar{x}})$ be a spread network with corresponding vector $\bar{\tau}$.
Define ADJ as a $|V_{\bar{x}}| \times |V_{\bar{x}}|$ matrix, where $\texttt{ADJ}(i, j) = 1$ if there exists a path from node i to node j on $G_{\bar{x}}$, and $\texttt{ADJ}(i, j) = 0$ otherwise.
for $t = 0$ to $T - 2$ **do**
  for all nodes $k \in V_{\bar{x}}^{T-t}$ **do**
    Initialize L_k as an array of all nodes $j \in V^-(k)$ such that $\bar{\tau}_j < T - t$.
    Sort nodes in L_k based on nonincreasing values of $\bar{\tau}$.
    for $p = 1$ to $|L_k|$ **do**
      Let j be the pth node in L_k.
      if $(j, k) \in A_{\bar{x}}$ **then**
        Set $q = |L_k|$, and let i be the qth node in L_k.
        Set flag `repeat = true`.
        while (`repeat == true`) **do**
          if $(i, k) \notin A_{\bar{x}}$ **then**
            if $\texttt{ADJ}(i, j) = 1$ **then**
              Remove arc (j, k) from $A_{\bar{x}}$ and add arc (i, k) to $A_{\bar{x}}$.
              Set `repeat = false`.
            end if
          else
            Set $q = q - 1$, and let i be the qth node in L_k.
            if $\bar{\tau}_i \geq \bar{\tau}_j$ **then**
              Set `repeat = false`.
            end if
          end if
        end while
      end if
    end for
  end for
end for

earlier than $\bar{\tau}_k$. The algorithm examines each arc $(j, k) \in A_{\bar{x}}$ in nonincreasing order of arc lengths (i.e., nodes j are considered in nonincreasing order of their $\bar{\tau}_j$-values). Algorithm 2 seeks a node $i \in L_k$, $(i, k) \notin A_{\bar{x}}$, having the smallest value of $\bar{\tau}_i$ such that there exists a path from node i to node j (i.e., such that $\texttt{ADJ}(i, j) = 1$). In case such node is found (with $\bar{\tau}_i < \bar{\tau}_j$), nodes i, j, and k satisfy the criteria of Theorem 4.4, and so we replace arc (j, k) with arc (i, k) in $A_{\bar{x}}$. This process repeats until no more arcs can be replaced.

Note that as Algorithm 2 proceeds, the spread network might be modified by "add" or "remove" operations performed in Step 15. However, the elements of matrix ADJ are never updated throughout the execution of Algorithm 2. This is due to the fact that $\texttt{ADJ}(i, j)$ correctly indicates the existence of a path between nodes i and j whenever it is examined in Step 14, even though the spread network might have been modified in earlier stages of Algorithm 2. To see this, suppose that at some stage of Algorithm 2, the value of $\texttt{ADJ}(i, j)$ is examined while visiting node $k \in V_{\bar{x}}$ in the for-loop at Step 4. Because this for-loop examines candidate nodes k in nonincreasing order of their $\bar{\tau}$-values, Algorithm 2 could have only modified the spread network by adding arcs (i', k') for some node $i' \in V_{\bar{x}}$ and $k' \in V_{\bar{x}}^t,\ t \geq \bar{\tau}_k$, or removing arcs (j', k') for some node $j' \in V_{\bar{x}}$ and $k' \in V_{\bar{x}}^t,\ t \geq \bar{\tau}_k$. Recall that the spread network contains no arcs (u, v) such that $\bar{\tau}_u \geq \bar{\tau}_v$. Therefore, the addition or deletion of arcs in Step 15 cannot create a new path, or disconnect an existing path, from node i to node j.

To analyze the complexity of Algorithm 2, observe that the construction of matrix ADJ takes $\mathcal{O}(Q|V_{\bar{x}}|^2)$ steps. For each node k examined in the for-loop at Steps 3 and 4, Algorithm 2 performs one sorting operation (Step 6), which is $\mathcal{O}(|V_{\bar{x}}| \log |V_{\bar{x}}|)$. For each node j examined in the for-loop in Step 7, Algorithm 2 executes $\mathcal{O}(|V_{\bar{x}}|)$ operations in the while-loop at Step 12 corresponding to each candidate node i. (Note that an arc (i, k) that is added to $A_{\bar{x}}$ after removing some arc (j, k) might be replaced later by some other arc (i', k) as the algorithm proceeds, which implies that a total of $\mathcal{O}(|V_{\bar{x}}|)$ nodes might be examined in the for-loop in Step 7). Therefore, for each node k examined in the for-loops at Steps 3 and 4, Algorithm 2 performs $\mathcal{O}(|V_{\bar{x}}|^2)$ operations, and hence, the overall complexity of Algorithm 2 is $\mathcal{O}(Q|V_{\bar{x}}|^2 + |V_{\bar{x}}|^3)$. In fact, the complexity can be more specifically stated as $\mathcal{O}(|V_{\bar{x}}|^3)$: If $Q \leq |V_{\bar{x}}|$, then this is obviously true, and if $Q > |V_{\bar{x}}|$, then every node in the spread network was influenced at time 0, $A_{\bar{x}}$ would necessarily be empty, and the algorithm would terminate in constant time.

4.4 Attacker's Problem Solution Approach

In order to generate the valid inequalities introduced in Section 4.3, we must solve the (NP-hard) attacker's problem. Therefore, the efficiency of our solution method is highly dependent on the time required to solve instances of the attacker's problem. This motivates further investigation of the attacker's problem with the aim of devising alternative formulations that can be more efficiently solved by mathematical optimization techniques.

Note that once the y^0-variables are fixed, the optimal value of each y_i^t-variable for $t \in \mathcal{T}$ can be readily determined via a polynomial-time procedure, which starts from time 1 and identifies the number of influenced nodes adjacent to node i at time 0. Then, $y_i^1 = 1$ if $x_i = 0$ and at least Q nodes adjacent to node i are influenced at time 0, and $y_i^1 = 0$ otherwise. By repeating the same operation for all other time periods, the optimal value of each y_i^t-variable will be either zero or one.

This observation suggests that a mathematical programming formulation for the attacker's problem that includes only $|V|$ binary variables may be attainable. However, the Constraint 4–2f cannot be relaxed in Problem 4–2, which indeed requires $\mathcal{O}(T|V|)$ binary variables. In this section, we investigate two alternative formulations for the attacker's problem that allow us to relax the binariness restriction on y_i^t-variables for $t \in \mathcal{T}$.

4.4.1 Reformulation 1: Exponential Set Model

In Section 4.4.1.1, we propose a reformulation for Problem 4–2 that requires $O(T)$ binary variables. In Section 4.4.1.2, we demonstrate how to efficiently implement Benders' decomposition to solve this formulation.

4.4.1.1 Model

For each $i \in V$, define $\mathcal{S}_i = \{S : S \subset V^-(i),\ |S| = |V^-(i)| - (Q-1)\}$. The following is a reformulation for problem 4–2.

$$\textbf{ATT2}(\bar{x}) : z(\bar{x}) = \max \sum_{i \in V} \left(r_i^0 y_i^0 + \sum_{t=1}^{T} r_i^t (y_i^t - y_i^{t-1}) \right) \tag{4–18a}$$

$$\text{s.t. } y_i^t \leq y_i^0 + \sum_{j \in S} y_j^{t-1} \qquad \forall i \in V,\ t \in \mathcal{T},\ S \in \mathcal{S}_i \tag{4–18b}$$

$$y_i^t \in \{0,1\} \qquad \forall i \in V,\ t \in \mathcal{T} \tag{4–18c}$$

$$\text{Constraints (4–2b), (4–2d), and (4–2e).} \tag{4–18d}$$

Note that the only difference between models 4–2 and 4–18 lies in the constraints that govern the spread of influence. According to Constraints 4–18b, if node $i \in V$ is initially influenced, then the RHS of Constraints 4–18b will be at least one for all $t \in \mathcal{T}$ and $S \in \mathcal{S}_i$, implying that node i will remain influenced at all time periods. Now, suppose that node $i \in V$ is not initially influenced, and examine Constraints 4–18b at time $t \in \mathcal{T}$. If fewer than Q nodes adjacent to node i are influenced at time $t-1$, then there exists a subset $\bar{S} \in \mathcal{S}_i$ such that no node in $\bar{S}$ is influenced at time period $t-1$. In this case, $y_i^t = 0$ due to Constraint 4–18b corresponding to $\bar{S}$. Otherwise, the RHS of Constraints 4–18b for all $S \in \mathcal{S}_i$ will be at least one for node i at time period t, and hence, $y_i^t = 1$ at optimality if $\bar{x}_i = 0$. The following theorem demonstrates that Constraint 4–18c can equivalently be relaxed to take continuous values.

Theorem 4.5. *Consider Problem ATT2($\bar{x}$) for any $\bar{x} \in X$, in which Constraints 4–18c are replaced with $0 \leq y_i^t \leq 1,\ \forall i \in V,\ t \in \mathcal{T}$. There exists an optimal solution $(\hat{y}^0, \ldots, \hat{y}^T)$ to this relaxation in which $\hat{y}_i^t \in \{0,1\},\ \forall i \in V,\ t \in \mathcal{T}$.*

Proof. Consider any feasible solution $\bar{y} = (\bar{y}^0, \ldots, \bar{y}^T)$ to the relaxed version of ATT2($\bar{x}$) in which Constraints 4–18c are replaced with $0 \leq y_i^t \leq 1,\ \forall i \in V,\ t \in \mathcal{T}$, and suppose that $0 < \bar{y}_i^t < 1$ for some $i \in V,\ t \in \mathcal{T}$. Let t' be the smallest time period for which $0 < \bar{y}_{i'}^{t'} < 1$ for some $i' \in V$. Consider the solution $\hat{y}$, which is identical to $\bar{y}$ with

the exception of setting $\hat{y}_{i'}^{t'} = 1$. First, note that $\hat{y}$ does not violate Constraints 4–2b, 4–2d, 4–2e, and the relaxed version of 4–18c. Also, our definition of t' implies that the RHS of Constraints 4–18b for node i', time t', and all $S \in \mathcal{S}_i$, must be at least one, and thus, increasing $y_{i'}^{t'}$ from $\bar{y}_{i'}^{t'}$ does not violate these constraints. Additionally, the RHS of all Constraints 4–18b corresponding to time $t' + 1$ will not decrease when $y_{i'}^{t'}$ increases. Hence, $\hat{y}$ must also be feasible to Problem 4–18. Finally, each y_i^t-variable has a nonnegative objective coefficient $r_i^t - r_i^{t+1}$. It follows that $\hat{y}$ cannot yield a worse objective value than $\bar{y}$. By repeating the same approach for all $y_i^t \in (0,1),\ i \in V,\ t \in \mathcal{T}$, we obtain a feasible solution in which $y_i^t \in \{0,1\},\ \forall i \in V,\ t \in \mathcal{T}$, with an objective value not worse than the objective value for $\bar{y}$. This completes the proof. □

Using Theorem 4.5, we henceforth relax Constraint 4–18c to $0 \leq y_i^t \leq 1,\ \forall i \in V,\ t \in \mathcal{T}$, in ATT2($\bar{x}$).

4.4.1.2 Benders' decomposition

In this section, we investigate the application of Benders' decomposition in solving model 4–18. Observe that model 4–18 reduces to the following linear program for given vectors $\bar{x}$ and $\bar{y}^0$:

$$\max \quad \sum_{i \in V} \left(\sum_{t=1}^{T-1} (r_i^t - r_i^{t+1}) y_i^t + r_i^T y_i^T \right) \tag{4–19a}$$

$$\text{s.t.} \quad y_i^1 \leq \bar{y}_i^0 + \sum_{j \in S} \bar{y}_j^0 \qquad \forall i \in V,\ S \in \mathcal{S}_i \tag{4–19b}$$

$$y_i^t - \sum_{j \in S} y_j^{t-1} \leq \bar{y}_i^0 \qquad \forall i \in V,\ t \in \mathcal{T} \backslash \{1\},\ S \in \mathcal{S}_i \tag{4–19c}$$

$$y_i^t \leq 1 - \bar{x}_i \qquad \forall i \in V,\ t \in \mathcal{T} \tag{4–19d}$$

$$y_i^t \geq 0 \qquad \forall i \in V,\ t \in \mathcal{T}. \tag{4–19e}$$

Let $\pi_{i,S}^1$, $\pi_{i,S}^t$, and μ_i^t be the dual variables associated with Constraints 4–21b, 4–21c, and 4–19d, respectively. Defining $\mathcal{S}_{j,i} = \{S : S \in \mathcal{S}_j,\ i \in S\}$ as the set of all sets

$S \in \mathcal{S}_j$ that include node i, we obtain the dual problem to 4–19 given $\bar{x}$ and $\bar{y}^0$:

$$\min \quad \sum_{i \in V} \sum_{t=2}^{T} \sum_{S \in \mathcal{S}_i} \bar{y}_i^0 \pi_{i,S}^t + \sum_{i \in V} \sum_{S \in \mathcal{S}_i} (\bar{y}_i^0 + \sum_{j \in S} \bar{y}_j^0) \pi_{i,S}^1 + \sum_{i \in V} \sum_{t=1}^{T} (1 - \bar{x}_i) \mu_i^t \tag{4–20a}$$

$$\text{s.t.} \quad \sum_{S \in \mathcal{S}_i} \pi_{i,S}^t - \sum_{j \in V^+(i)} \sum_{S \in \mathcal{S}_{j,i}} \pi_{j,S}^{t+1} + \mu_i^t \geq r_i^t - r_i^{t+1} \qquad \forall i \in V,\ t \in \mathcal{T} \backslash \{T\} \tag{4–20b}$$

$$\sum_{S \in \mathcal{S}_i} \pi_{i,S}^T + \mu_i^T \geq r_i^T \qquad \forall i \in V \tag{4–20c}$$

$$\pi_{i,S}^t \geq 0 \qquad \forall i \in V,\ S \in \mathcal{S}_i,\ t \in \mathcal{T} \tag{4–20d}$$

$$\mu_i^t \geq 0 \qquad \forall i \in V,\ t \in \mathcal{T}. \tag{4–20e}$$

Note that an optimal solution must exist to Problem 4–20, because the objective function value is always nonnegative, and a feasible solution can be obtained by setting $\mu_i^t = r_i^t - r_i^{t+1},\ \forall i \in V,\ t \in \mathcal{T}$, with all π-variables equal to zero. Letting Ξ denote the set of all extreme points to Problem 4–20, the Benders' master problem is given as:

$$\max \quad \psi \tag{4–21a}$$

$$\text{s.t.} \quad \psi \leq \sum_{i \in V} (r_i^0 - r_i^1) y_i^0 + \sum_{i \in V} \sum_{S \in \mathcal{S}_i} \bar{\pi}_{i,S}^1 (y_i^0 + \sum_{j \in S} y_j^0) + \sum_{i \in V} \sum_{t=2}^{T} \sum_{S \in \mathcal{S}_i} \bar{\pi}_{i,S}^t y_i^0 + \sum_{i \in V} \sum_{t \in \mathcal{T}} (1 - \bar{x}_i) \bar{\mu}_i^t \qquad \forall (\bar{\pi}, \bar{\mu}) \in \Xi \tag{4–21b}$$

$$y^0 \in Y(\bar{x}), \tag{4–21c}$$

with Problem 4–19 being the Benders' subproblem. The restricted master problem (RMP) is given by 4–21 with only a limited set of dual extreme points, denoted by $\bar{\Xi}$, and corresponding Constraints 4–21b.

Note, however, that Problem 4–19 has an exponential number of Constraints 4–21c. Therefore, we aim to solve this subproblem using methods other than linear programming techniques.

Let $\bar{y} = (\bar{y}^1, \ldots, \bar{y}^T)$ be an optimal solution to Problem 4–19, and suppose $\bar{y}_i^t = 0$ for some unprotected node $i \in V$ and time $t \in \mathcal{T}$. Then, there must exist some $\bar{S}_i^t \in \mathcal{S}_i$ such

that no node in $\bar{S}_i^t$ is influenced at time $t-1$. We refer to $\bar{S}_i^t$ as a *safe subset* for node $i \in V$ and time $t \in \mathcal{T}$. It is worth noting that a safe subset for node i such that $y_i^{t'} = 0$ is a safe subset for all time $t \leq t'$. We will thus discard the t-index from $\bar{S}_i^t$, and simply refer to $\bar{S}_i$ as a safe subset for node i for all time periods t such that $y_i^t = 0$. We provide a dual recovery algorithm (DRA) for identifying an optimal solution to Problem 4–20 as follows.

Step 1 For all $i \in V$, if $\bar{x}_1 = 1$ or $y_i^T = 1$, then set $\bar{\mu}_i^T = r_i^T$ and $\bar{\pi}_{i,S}^T = 0,\ \forall S \in \mathcal{S}_i$. Otherwise, set $\bar{\mu}_i^T = 0$, $\bar{\pi}_{i,\bar{S}_i}^T = r_i^T$, and $\bar{\pi}_{i,S}^T = 0,\ \forall S \in \mathcal{S}_i \backslash \{\bar{S}_i\}$. Initialize $t = T$.

Step 2 If $t = 0$, then terminate. Otherwise, set $t = t-1$ and proceed to Step 3.

Step 3 For every $i \in V$:

a. If $\bar{x}_i = 1$, then $\bar{\mu}_i^t = r_i^t - r_i^{t+1} + \sum_{j \in V^+(i)} \sum_{S \in \mathcal{S}_{j,i}} \bar{\pi}_{j,S}^{t+1}$ and $\bar{\pi}_{i,S}^t = 0,\ \forall t \in \mathcal{T} \backslash \{T\},\ S \in \mathcal{S}_i$.

b. If $\bar{x}_i = 0$ and $\bar{y}_i^t = 0$, then set $\bar{\mu}_i^t = 0$, $\bar{\pi}_{i,\bar{S}_i}^t = r_i^t - r_i^{t+1} + \sum_{j \in V^+(i)} \sum_{S \in \mathcal{S}_{j,i}} \bar{\pi}_{j,S}^{t+1}$, and $\bar{\pi}_{i,S}^t = 0,\ \forall S \in \mathcal{S}_i \backslash \{\bar{S}_i\}$.

c. If $\bar{x}_i = 0$ and $\bar{y}_i^t = 1$, then set $\bar{\mu}_i^t = r_i^t - r_i^{t+1}$ and $\bar{\pi}_{i,S}^t = 0,\ \forall S \in \mathcal{S}_i$.

Proceed to Step 2.

In the following lemma, we establish the optimality of the solution identified by DRA.

Lemma 9. *Suppose that the solution $\bar{y} = (\bar{y}^1, \ldots, \bar{y}^T)$ is optimal to Problem 4–19. Let $\bar{S}_i$ be a safe subset for each unprotected node $i \in V$ such that $y_i^t = 0$ for some $t \in \mathcal{T}$. Then, DRA constructs an optimal solution $(\bar{\pi}, \bar{\mu})$ to Problem 4–20.*

Proof. First, we show that $(\bar{\pi}, \bar{\mu})$ as constructed by DRA is feasible to Problem 4–20. Let $t = T$. Note that the left-hand-side (LHS) of Constraints 4–20c corresponding to any node $i \in V$ will be r_i^T from Step 1. Therefore, $(\bar{\pi}, \bar{\mu})$ satisfies Constraints 4–20c. Next, let $t = T-1$. From Step 3a, the LHS of Constraint 4–20b for protected node $i \in V$ and time $T-1$ will be $r_i^{T-1} - r_i^T$. Similarly, for each unprotected node $i \in V$ such that $y_i^{T-1} = 0$, Step 3b guarantees that the LHS of Constraint 4–20b corresponding to node i and time $T-1$ will equal $r_i^{T-1} - r_i^T$. Now, consider any influenced node $i \in V$

at time $T-1$, i.e., $\bar{y}_i^t = 1$. Note that node i cannot be in the safe subset of any node $j \in V^+(i)$ and time T, i.e., $\sum_{j \in V^+(i)} \sum_{S \in \mathcal{S}_{j,i}} \bar{\pi}_{j,S}^{t+1} = 0$ in the corresponding Constraint 4–20b. Therefore, by setting $\bar{\mu}_i^{T-1} = r_i^{T-1} - r_i^T$ and $\bar{\pi}_{i,S}^t = 0,\ \forall S \in \mathcal{S}_i$, in Step 3c, Constraint 4–20b is satisfied. Hence, the solution $(\bar{\pi}, \bar{\mu})$ satisfies Constraints 4–20b. By inductively repeating a similar argument for all nodes $i \in V$ and all times $T-2, \ldots, 1$, we conclude that $(\bar{\pi}, \bar{\mu})$ is feasible to 4–20.

Next, we must show that the dual objective computed at $(\bar{\pi}, \bar{\mu})$ matches the optimal value of the primal objective function, which is given by $\sum_{i \in V: \bar{x}_i = 0} r_i^{\bar{\tau}_i}$. Consider any node $i \in V$. If $\bar{y}_i^0 = 1$, then $\bar{y}_i^t = 1,\ \forall t \in \mathcal{T}$, implying that $\bar{\pi}_{i,S}^t = 0,\ \forall t \in \mathcal{T},\ S \in \mathcal{S}_i$. Hence, we obtain:

$$\sum_{i \in V} \sum_{t=2}^{T} \sum_{S \in \mathcal{S}_i} \bar{y}_i^0 \bar{\pi}_{i,S}^t = 0.$$

Also, note that if $\bar{\pi}_{i,S}^1 > 0$ for some $S \in \mathcal{S}_i$, then S must be a safe subset for node i and time $t = 1$ implying that $\bar{y}_i^0 = 0$ and $\bar{y}_j^0 = 0, \forall j \in S$. Hence,

$$\sum_{i \in V} \sum_{S \in \mathcal{S}_i} (\bar{y}_i^0 + \sum_{j \in S} \bar{y}_j^0) \bar{\pi}_{i,S}^1 = 0.$$

Thus, the objective function 4–20a evaluates to $\sum_{i \in V} \sum_{t \in \mathcal{T}} (1 - \bar{x}_i) \bar{\mu}_i^t$ at $(\bar{\pi}, \bar{\mu})$. From Steps 1, 3b, and 3c, this term reduces to the primal optimal objective function value, i.e., $\sum_{i \in V: \bar{x}_i = 0} r_i^{\bar{\tau}_i}$. This completes the proof. □

An immediate result from Lemma 9 is that we only need to identify a single safe subset $\bar{S}_i$ for each unprotected node $i \in V$ that is not influenced at time 1. This allows us to discard S-indices from the π-variables when referring to Problem 4–20, and to rewrite Constraints 4–21b as:

$$\psi \leq \sum_{i \in V} (r_i^0 - r_i^1) y_i^0 + \sum_{i \in V} \bar{\pi}_i^1 (y_i^0 + \sum_{j \in \bar{S}_i} y_j^0) + \sum_{i \in V} \sum_{t=2}^{T} \bar{\pi}_i^t y_i^0 + \sum_{i \in V} \sum_{t \in \mathcal{T}} (1 - \bar{x}_i) \bar{\mu}_i^t,$$

or equivalently,

$$\psi \leq \sum_{i \in V} \left(r_i^0 - r_i^1 + \bar{\pi}_i^1 + \sum_{j:\, i \in \bar{S}_j} \bar{\pi}_j^1 + \sum_{t=2}^{T} \bar{\pi}_i^t \right) y_i^0 + \sum_{i \in V} \sum_{t \in \mathcal{T}} (1 - \bar{x}_i) \bar{\mu}_i^t. \tag{4–22}$$

Inequality 4–22 can be strengthened by a standard coefficient tightening procedure as follows. Let η_i be the coefficient of variable $y_i^0,\ i \in V$, in 4–22. The following inequality:

$$\psi \leq \sum_{i \in V} \left(\min\{\eta_i,\ z_{\max}(\bar{x}) - \sum_{i \in V} \sum_{t \in \mathcal{T}} (1 - \bar{x}_i) \bar{\mu}_i^t\} \right) y_i^0 + \sum_{i \in V} \sum_{t \in \mathcal{T}} (1 - \bar{x}_i) \bar{\mu}_i^t, \tag{4–23}$$

is valid to 4–21, because $\eta_i \geq 0$ and $\bar{y}_i^0 \in \{0, 1\},\ \forall i \in V$; $z_{\max}(\bar{x}) - \sum_{i \in V} \sum_{t \in \mathcal{T}} (1 - \bar{x}_i) \bar{\mu}_i^t \geq 0$ (noting that $\sum_{i \in V} \sum_{t \in \mathcal{T}} (1 - \bar{x}_i) \bar{\mu}_i^t$ is the optimal attacker's objective in the last inequality, which is no more than $z_{\max}(\bar{x})$); and $\psi \leq z_{\max}(\bar{x})$. Thus, when solving the Benders' master problem, we replace 4–21b with 4–23.

4.4.2 Reformulation 2: Compact Model

In this section, we provide an alternative compact (polynomial-size) formulation for the attacker's problem, in which the binary restrictions on the y_i^t-variables can be relaxed. For each $i \in V$, arbitrarily order the nodes in $V^-(i)$ as $\{i_1, \ldots, i_{|V^-(i)|}\}$. Define $v_{imk}^t = 1$ if at least k of first m nodes in $V^-(i)$ are influenced at time $t-1$, and $v_{imk}^t = 0$ otherwise. By convention, we let $v_{imk}^t = 0,\ k > m$. Letting $\mathbb{N}_p = \{1, \ldots, p\}$ for any positive

integer p, the following is a reformulation for model 4–2, given $\bar{x}$: a

$$\textbf{ATT3}(\bar{x}): \max \quad \sum_{i \in V} \left(r_i^0 y_i^0 + \sum_{t=1}^{T} r_i^t (y_i^t - y_i^{t-1}) \right) \tag{4–24a}$$

$$\text{s.t.} \quad v_{imk}^t \leq v_{i,m-1,k-1}^t \qquad \forall i \in V,\ t \in \mathcal{T},\ m \in \mathbb{N}_{|V^-(i)|},\ k \in \mathbb{N}_{\min\{m,Q\}} \tag{4–24b}$$

$$v_{imk}^t \leq v_{i,m-1,k}^t + y_{i_m}^{t-1} \quad \forall i \in V,\ t \in \mathcal{T},\ m \in \mathbb{N}_{|V^-(i)|},\ k \in \mathbb{N}_{\min\{m,Q\}} \tag{4–24c}$$

$$y_i^t \leq y_i^0 + v_{i,|V^-(i)|,Q}^t \qquad \forall i \in V,\ t \in \mathcal{T} \tag{4–24d}$$

$$y_i^t \leq 1 - \bar{x}_i \qquad \forall i \in V,\ t \in \mathcal{T} \tag{4–24e}$$

$$v_{imk}^t \in \{0,1\} \qquad \forall i \in V,\ t \in \mathcal{T},\ m \in \mathbb{N}_{|V^-(i)|},\ k \in \mathbb{N}_{\min\{m,Q\}} \tag{4–24f}$$

$$y_i^t \in \{0,1\} \qquad \forall i \in V,\ t \in \mathcal{T} \tag{4–24g}$$

$$y^0 \in Y(\bar{x}). \tag{4–24h}$$

The objective function 4–24a is the same as the objective function in model 4–2. Constraints 4–24b and 4–24c enforce the definition of v_{imk}^t. To see this, suppose that fewer than k of the first m nodes in $V^-(i)$ are influenced at time $t-1$. If node i_m is influenced at time $t-1$, then at most $k-2$ of first $m-1$ nodes in $V^-(i)$ can be influenced at time $t-1$. This implies that $v_{i,m-1,k-1}^t = 0$, and Constraints 4–24b force $v_{imk}^t = 0$. Otherwise, if i_m is not influenced at time $t-1$, then at most $k-1$ of the first $m-1$ nodes in $V^-(i)$ are influenced at time $t-1$, i.e., $v_{i,m-1,k}^t = 0$, and Constraints 4–24c force $v_{imk}^t = 0$. On the other hand, if at least k of the first m nodes in $V^-(i)$ are influenced at time $t-1$, then at least $k-1$ of the first $m-1$ nodes were influenced at time $t-1$. Furthermore, either at least k of the first $m-1$ nodes were influenced, or node i_m itself was influenced at time $t-1$. Hence, $v_{i,m-1,k-1}^t = 1$ and $v_{i,m-1,k-1}^t + y_{i_m}^{t-1} \geq 1$, which allows $v_{imk}^t \geq 1$ (as will be the case at optimality). Constraints 4–24d imply that node i cannot be influenced at time t unless either it has been initially influenced or at

least Q of its adjacent nodes are influenced at time $t-1$. The binariness of the v- and y-variables and the budget limit are enforced by Constraints 4–24f–4–24h. However, the following theorem demonstrates that Constraints 4–24f and 4–24g can equivalently be relaxed to take continuous values.

Theorem 4.6. *Consider the relaxed version of ATT3($\bar{x}$) in which Constraints 4–24f and 4–24g are replaced with the following constraints:*

$$0 \leq v_{imk}^t \leq 1 \qquad \forall i \in V,\ t \in \mathcal{T},\ m \in \mathbb{N}_{|V^-(i)|},\ k \in \mathbb{N}_{\min\{m,Q\}} \tag{4–25a}$$

$$0 \leq y_i^t \leq 1 \qquad \forall i \in V,\ t \in \mathcal{T}. \tag{4–25b}$$

There exists an optimal solution $(\hat{y}, \hat{v})$ to the relaxed problem in which $\hat{v}_{imk}^t \in \{0,1\},\ \forall i \in V,\ t \in \mathcal{T},\ m \in \mathbb{N}_{|V^-(i)|},\ k \in \mathbb{N}_{\min\{m,Q\}}$, and $\hat{y}_i^t \in \{0,1\},\ \forall i \in V,\ t \in \mathcal{T}$.

Proof. Consider an optimal solution $(\bar{y}, \bar{v})$ to the relaxation of ATT3($\bar{x}$) described in the theorem in which $\bar{y}_i^t \in (0,1)$ for some $i \in V,\ t \in \mathcal{T}$, and/or $\bar{v}_{imk}^t \in (0,1)$ for some $i \in V,\ t \in \mathcal{T},\ m \in \mathbb{N}_{|V^-(i)|},\ k \in \mathbb{N}_{\min\{m,Q\}}$. Let t' be the smallest index for which either $0 < \bar{y}_i^{t'} < 1$ for some $i \in V$, or $0 < \bar{v}_{imk}^{t'} < 1$ for some $i \in V,\ m \in \mathbb{N}_{|V^-(i)|},\ k \in \mathbb{N}_{\min\{m,Q\}}$. We discuss two cases:

- Case 1: t' is defined for a variable $y_i^{t'}$. Consider the solution $(\hat{y}, \hat{v})$, which is identical to $(\bar{y}, \bar{v})$ but with $\hat{y}_i^{t'} = 1$. Note that increasing the value of $y_i^{t'}$ from $\bar{y}_i^{t'}$ serves to relax Constraints 4–24c. Moreover, the RHS of Constraints 4–24d and 4–24e corresponding to node i and t' must be at least one, or else $\bar{y}_i^{t'}$ would have equalled zero. Hence, the solution $(\hat{y}, \hat{v})$ is feasible to model 4–24a. Also, the objective value for $(\hat{y}, \hat{v})$ cannot be worse than that for $(\bar{y}, \bar{v})$, because the objective function coefficient for $y_i^{t'}$ is nonnegative.

- Case 2: t' is defined for a variable $v_{imk}^{t'}$. Let m' be the smallest index for which $0 < \bar{v}_{im'k}^{t'} < 1$ for some $i \in V$ and $k \in \mathbb{N}_{\min\{m,Q\}}$, and let k' be the smallest index for which $0 < \bar{v}_{im'k'}^{t'} < 1$ for some $i \in V$. Consider the solution $(\hat{y}, \hat{v})$, which is identical to $(\bar{y}, \bar{v})$ but with $\hat{v}_{im'k'}^{t'} = 1$. Increasing $v_{im'k'}^{t'}$ from $\bar{v}_{im'k'}^{t'}$ relaxes Constraints 4–24d. Moreover, note that our definitions of m' and k' imply that the RHS of Constraints 4–24b and 4–24c corresponding to node i, time t', index m' and index k' are at least one. Therefore increasing $v_{im'k'}^{t'}$ from $\bar{v}_{im'k'}^{t'}$ will not violate any of Constraints 4–24b and 4–24c. Hence, the solution $(\hat{y}, \hat{v})$ is feasible to model 4–24a. Finally, note that both $(\bar{y}, \bar{v})$ and $(\hat{y}, \hat{v})$ yield the same objective function value in 4–24a.

Table 4-1. Size comparison of attacker's problem formulations.

Model	Binary Variables	Continuous Variables	Constraints
ATT1	$\mathcal{O}(T\|V\|)$	0	$\mathcal{O}(T\|V\|)$
ATT2	$\mathcal{O}(\|V\|)$	$\mathcal{O}(T\|V\|)$	$\mathcal{O}(T\|V\|\binom{\|V\|}{Q})$
ATT3	$\mathcal{O}(\|V\|)$	$\mathcal{O}(TQ\|V\|^2)$	$\mathcal{O}(TQ\|V\|^2)$

We repeat the argument given above until all fractional y- and v-variable values become binary. The solution identified at the end of this process remains feasible and has an objective function value that is at least as large as that for $(\bar{y}, \bar{v})$. This completes the proof. □

In the remainder of the chapter, we thus replace 4–24f and 4–24g with 4–25a and 4–25b, respectively. Note that while only $|V|$ binary variables are needed in model 4–24a, the formulation requires the addition of $\mathcal{O}(TQ|V|^2)$ continuous variables. Table 4-1 compares the size of the proposed three formulations for the attacker's problem.

4.5 Computational Results

In this section, we study the performance of our proposed methods on randomly generated test instances. We start by introducing the parameters used to generate the test instances and discussing the implementation details in Section 4.5.1. In Section 4.5.2, we investigate the efficiency of employing CPLEX in solving the attacker's problem using formulations ATT1, ATT2, and ATT3. Finally, we examine the efficacy of CPA equipped with various cutting planes in Section 4.5.3.

4.5.1 Implementation Details

We implemented all algorithms in C++ equipped with CPLEX 12.3 Concert Technology on an IBM x3650 system with two Intel E5640 Xeon processors and 24 gigabytes of memory. For studying the methods for the attacker's problem, we set 600 seconds as the maximum allowable running time for CPLEX. We set the maximum running time to 1800 seconds when studying our cutting-plane algorithms for the defender's problem.

We consider two types of randomly generated networks: 1) networks with arbitrary degree distribution (denoted by ADD networks), and 2) scale-free networks (denoted by SF networks). While the degree distribution for ADD networks is determined by arbitrarily-chosen density values, SF networks are associated with power-law degree distributions [2]. SF networks are widely known for reflecting the topology of various real-world large-scale communication and social networks.

Table 4-2 shows the parameters and the corresponding values that we used to generate the test instances for both types of networks. All parameters are randomly generated as integers derived from a uniform distribution over the stated range, except for the budget. For the defender's and the attacker's budget, we define coefficients λ_l and λ_f, respectively, which are uniformly generated over the continuous interval $[0.35, 0.65]$. Next, we let $B = \lfloor \lambda_l S_b \rfloor$ and $D = \lfloor \lambda_f S_c \rfloor$, where S_b and S_c are the sum of all generated b- and c-values, respectively.

Table 4-2. Parameters used to generate test instances

Parameter Name	**Value**
Defender's protection cost (b)	[30, 100]
Attacker's initial attack cost (c)	[100, 200]
Infection rewards (r)	[70, 180]
Defender's budget coefficient (λ_l)	[0.35, 0.65]
Attacker's budget coefficient (λ_f)	[0.35, 0.65]

In order to generate (directed) SF network instances, we employ the $\tilde{\alpha}$-preferential attachment scheme suggested by Chung and Lu [21]. Their method starts with an initial graph G_0 formed by one vertex having one loop. At each step $s > 0$, G_s is constructed from G_{s-1} by adding a new node with an outgoing arc to an existing node in G_{s-1} with probability p_1, adding a new node with an incoming arc from an existing node in G_{s-1} with probability p_2, or adding a directed edge between two existing nodes in G_{s-1} with probability $1 - p_1 - p_2$. An existing node from G_{s-1} is chosen to be a tail (or a head) node based on a probability proportional to sum of the number of the node's outgoing (or incoming) arcs and a chosen parameter $\tilde{\alpha}$. The process stops when the desired

number of nodes exists in G_s. Note that smaller values of $p_1 + p_2$ result in graphs having lower density. For our computational study, we choose the values of all other parameters according to the values stated in Table 4-2.

Note that CPA relies on solving possibly many instances of the attacker's problem. Therefore, it is crucial to pick the most efficient solution approach to solve the attacker's problem. In order to compare the computational efficiency of various attacker's formulations discussed in Sections 4.2 and 4.4, we solve models 4–2 (denoted by ATT1) and 4–18 (denoted by ATT3) directly using CPLEX. We solve ATT2 using the Benders' decomposition method presented in Section 4.4.1.2, where we compute $z_{\max}(\bar{x})$ (used in 4–23) by solving Problem 4–5.

For the defender's problem, we consider six variants of CPA. For the first variant, denoted by CPA1, we implement CPA equipped with valid inequality 4–10. Next, we consider different variants of CPA, denoted by CPA2-1, CPA2-2, CPA2-3, and CPA2-4, in which we use valid inequalities 4–12. These variants differ in the way that the undirected acyclic graph $\mathcal{G}$ (stated in Theorem 4.2) is constructed. For CPA2-1, we let $\mathcal{G}$ be a star graph for which we randomly pick a center node. CPA2-2 is the implementation in which $\mathcal{G}$ is constructed as a random spanning tree. For CPA2-3 and CPA2-4, we construct acyclic subgraphs that have $\lceil |V_{\bar{x}}|/2 \rceil$ and $\lceil |V_{\bar{x}}|/10 \rceil$ randomly selected edges, respectively. Finally, we investigate a variant of CPA, denoted by CPA3, which is equipped with valid inequality 4–13. In particular, at each iteration of CPA3, we arbitrarily construct a spread network to derive its corresponding valid inequality 4–13, and employ Algorithm 2 to strengthen the inequalities.

Finally, recall that model 4–8 can be solved to compute an upper bound for the optimal objective value of the defender's problem. We also proposed a greedy algorithm for this purpose, but initial computational experiments indicate that the time required to solve Problem 4–8 is insignificant compared to the overall time required by CPA variants.

Hence, all CPA variants compute $z_{\max}$ by solving Problem 4–8 rather than employing the proposed heuristic.

4.5.2 Results for the attacker's problem

For our computational study of the attacker's problem, we start by examining instances generated based on ADD networks. We consider two scenarios for these instances. In the first scenario, we study the attacker's problem exactly as stated in Section 4.2. For the second scenario, we investigate the case in which some nodes cannot be attacked at time zero, but can possibly become influenced after time zero. These nodes are vulnerable to attack, but not directly accessible to the attacker (for each such node $i \in V$, we simply fix $y_i^0 = 0$).

For the first scenario, we start by considering eight value sets for parameters $|V|$, T, and Q. Moreover, we consider three graph density values, d, as 0.05, 0.2, and 0.4, resulting in 24 instance sets. Finally, we generate ten instances for each set. Table 4-3 illustrates the average time (in seconds) required by each implementation to solve test instances, where a time of 600 seconds is recorded for each instance that does not solve within the computational limits. For any combination of $|V|$, T, Q, and d in which the algorithm cannot identify an optimal solution within 600 seconds for at least one instance, we also record the average optimality gap produced by the algorithm over all such instances.

According to Table 4-3, ATT1 outperforms ATT2 and ATT3. Note that the instances with $d = 0.05$ are significantly more difficult to solve than cases for which $d = 0.2$ or $d = 0.4$. ATT2 is generally outperformed by the other two variants on these instances. This stems from the fact that the reduction in computational time due to the DRA method cannot compensate for the time required to solve the Benders' master problem. Finally, it is worth noting that the efficacy of ATT3 significantly declines for larger instances having denser graphs. One possible reason for this behavior is due to the fact that the number of Constraints 4–24b and 4–24c significantly increases for denser graphs,

resulting in longer running times for ATT3. As a result, ATT3 is also outperformed by ATT2 on instances with $d = 0.4$.

Table 4-3. Computational results for the first scenario of the attacker's problem on ADD networks

		ATT1		ATT2		ATT3	
Set ($\|V\|$, T, Q)	d	Avg Time	Avg Gap	Avg Time	Avg Gap	Avg Time	Avg Gap
(25, 2, 3)	0.05	0.03	0	0.05	0	0.03	0
	0.2	0.08	0	6.67	0	0.09	0
	0.4	0.03	0	0.04	0	0.07	0
(50, 3, 4)	0.05	0.29	0	200.54	1.3%	0.28	0
	0.2	0.08	0	64.21	1.0%	0.57	0
	0.4	0.03	0	0.02	0	0.36	0
(75, 4, 5)	0.05	2.97	0	528.99	5.7%	2.27	0
	0.2	0.14	0	180.25	1.0%	6.03	0
	0.4	0.05	0	0.04	0	1.79	0
(100, 5, 6)	0.05	196.57	2.0%	600	14.3%	203.84	1.5%
	0.2	0.10	0	4.96	0	4.91	0
	0.4	0.08	0	0.08	0	4.45	0
(125, 6, 8)	0.05	475.32	10.9%	600	26.0%	480.98	7.0%
	0.2	0.11	0	23.08	3.0%	8.39	0
	0.4	0.11	0	0.15	0	13.5	0
(150, 7, 9)	0.05	600	12.7%	600	31.8%	600	13.5%
	0.2	0.27	0	60.91	10.1%	76.24	0
	0.4	0.15	0	0.23	0	22.89	0
(175, 7, 10)	0.05	600	24.7%	600	38.1%	600	28.6%
	0.2	0.20	0	55.61	1.1%	89.80	0
	0.4	0.24	0	0.35	0	63.07	0
(200, 8, 10)	0.05	600	18.1%	600	34.6%	600	20.0%
	0.2	0.21	0	0.56	0	47.42	0
	0.4	1.93	0	0.97	0	579.51	30.2%

In order to generate test instances for the second scenario, we consider four value sets for parameters $|V|$, T, Q, and d. Let σ be the ratio of the number of nodes that cannot be initially attacked to $|V|$. By varying $\sigma \in \{40\%, 70\%\}$, we generate a total of eight sets, each having ten randomly generated instances. Table 4-4 reports the results of this experiment using the same column definitions as in Table 4-3.

Recall that ATT2 is designed to combat the growth of mathematical programming models as a function of T, wherein the DRA method executes a low-polynomial-time

Table 4-4. Computational results for the second scenario of the attacker's problem on ADD networks

		ATT1		ATT2		ATT3	
Set ($\|V\|$, T, Q, d)	σ	Avg Time	Avg Gap	Avg Time	Avg Gap	Avg Time	Avg Gap
(100, 75, 9, 0.05)	40%	0.62	0	60.38	0	3.44	0
	70%	0.28	0	0.03	0	2.8	0
(150, 100, 11, 0.05)	40%	61.09	0	300.04	1.8%	69.256	0
	70%	0.08	0	0.05	0	8.637	0
(200, 125, 13, 0.05)	40%	2.85	0	480.06	7.6%	24.45	0
	70%	2.57	0	0.11	0	20.98	0
(250, 175, 15, 0.05)	40%	8.57	0	480.47	4.0%	138.09	1.5%
	70%	7.36	0	0.321	0	87.11	0

routine to calculate the impact of an attacker's action and generate a Benders' cut. The tradeoff is that ATT2 requires the solution of a (mixed-integer) master problem that may require the addition of many cuts. We observe that when $\sigma = 40\%$, ATT1 still outperforms the other two variants. However, ATT2 outperforms ATT1 and ATT3 for $\sigma = 70\%$. Evidently, when σ changes from 40% to 70%, the Benders' master problem becomes less difficult to solve, and solving ATT2 becomes the most efficient approach.

We also study the attacker's problem for the SF network instances. We start by considering ten value sets for parameters $|V|$, T, and Q. In order to generate relatively dense and sparse SF networks, we also consider two value sets for the probability values, p_1 and p_2, resulting in 20 instance sets. Finally, we generate ten instances for each set. Table 4-5 illustrates the average time (in seconds) required by each implementation to solve test instances. Note that we have not reported the time and gap information for ATT2 on instances having 3000 or more nodes, because ATT2 is clearly inferior to ATT1 and ATT3 on these instances.

According to Table 4-5, ATT1 outperforms ATT3, with ATT2 being an impractical method to solve this class of instances. The SF network instances are, in general, very sparse compared to the ADD network instances. As a result, ATT1 and ATT3 are able to solve larger instances of the attacker's problem on SF networks compared to ADD

Table 4-5. Computational results for the attacker's problem on SF networks

		ATT1		ATT2		ATT3	
Set ($\|V\|$, T, Q)	(p_1, p_2)	Avg Time	Avg Gap	Avg Time	Avg Gap	Avg Time	Avg Gap
(250, 3, 2)	(0.2, 0.2)	0.15	0	600.1	11%	0.17	0
	(0.4, 0.4)	0.12	0	600.0	10%	0.15	0
(500, 4, 3)	(0.2, 0.2)	0.15	0	600.1	19%	0.28	0
	(0.4, 0.4)	0.18	0	600.1	17%	0.30	0
(750, 5, 3)	(0.2, 0.2)	0.28	0	600.3	19%	0.82	0
	(0.4, 0.4)	0.23	0	600.0	19%	0.53	0
(1000, 5, 4)	(0.2, 0.2)	0.34	0	600.2	21%	0.77	0
	(0.4, 0.4)	0.27	0	600.1	18%	0.60	0
(3000, 8, 7)	(0.2, 0.2)	3.65	0	-	-	20.72	0
	(0.4, 0.4)	3.62	0	-	-	12.7	0
				-	-		
(5000, 9, 8)	(0.2, 0.2)	12.16	0	-	-	97.91	0
	(0.4, 0.4)	12.21	0	-	-	42.33	0
				-	-		
(7000, 10, 8)	(0.2, 0.2)	28.75	0	-	-	116.48	0
	(0.4, 0.4)	29.88	0	-	-	71.76	0
				-	-		
(9000, 10, 9)	(0.2, 0.2)	58.14	0	-	-	216.59	0
	(0.4, 0.4)	59.25	0	-	-	201.8	0
				-	-		
(11000, 11, 9)	(0.2, 0.2)	97.22	0	-	-	286.43	0
	(0.4, 0.4)	83.83	0	-	-	213.92	0
				-	-		
(13000, 12, 10)	(0.2, 0.2)	171.05	0	-	-	449.02	0
	(0.4, 0.4)	170.72	0	-	-	392.76	0
				-	-		
(15000, 13, 11)	(0.2, 0.2)	374.65	0	-	-	718.7	1%
	(0.4, 0.4)	364.53	0	-	-	717.26	1%

networks. Note that while ATT1 spends roughly the same amount of time on dense SF networks ($p_1 = p_2 = 0.2$) and sparse SF networks ($p_1 = p_2 = 0.4$), ATT3 requires significantly less time in solving sparse SF networks. This behavior is due to the fact that the number of constraints in ATT1 does not depend on the sparsity of the network, whereas ATT3 requires fewer Constraints 4–24d for sparse SF networks.

4.5.3 Results for the defender's problem

For the defender's problem, we only generate ADD network instances. (Our preliminary computational study on instances of the defender's problem on SF

networks did not lead to significantly different results compared to the instances that are generated on ADD networks.)

Recall that Step 2 of CPA requires the solution of the attacker's problem given a defender's decision vector. Based on the results from Section 4.5.2, we employ formulation ATT1 to solve the attacker's problem. We generate six value sets for parameters $|V|$, T, and Q. For each of the value sets, we consider three density values of 0.05, 0.2, and 0.4. Finally, we generate ten instances for each set. In Table 4-6, we report the average time (in seconds) required by each implementation to solve test instances, as well as the average optimality gap produced by the algorithm. Similar to Table 4-3, we compute the average optimality gap over all instances that were not solved within 1800 seconds.

Table 4-6. Computational results of CPA implementations

		CPA1		CPA2-1		CPA2-2		CPA2-3		CPA2-4		CPA3	
Set ($\|V\|$, T, Q)	d	Time	Gap	Time	Gap	Time	Gap	Time	Gap	Time	Gap	Time	Gap
(15, 4, 3)	0.05	55.4	0	8.5	0	1.8	0	1.8	0	1.9	0	1.6	0
	0.2	58.0	0	4.9	0	1.4	0	1.5	0	1.6	0	1.4	0
	0.4	122.3	0	8.1	0	2.7	0	2.7	0	2.6	0	2.4	0
(17, 5, 4)	0.05	169.5	0	13.7	0	5.1	0	4.7	0	5.1	0	4.1	0
	0.2	268.6	0	11.7	0	4.2	0	3.9	0	3.9	0	3.4	0
	0.4	490.9	0	22.8	0	9.6	0	9.4	0	9.7	0	8.2	0
(19, 6, 5)	0.05	1074.1	20.7%	48.6	0	22.4	0	21.6	0	21.5	0	16.9	0
	0.2	1190.2	20.4%	44.7	0	19.8	0	20.3	0	20.1	0	15.6	0
	0.4	1723.4	21.3%	74.2	0	35.9	0	33.2	0	34.9	0	25.2	0
(21, 7, 6)	0.05	1754.6	19.8%	132.9	0	75.8	0	88.1	0	72.4	0	57.1	0
	0.2	1754.7	22.4%	203.6	0	121.9	0	116.7	0	122.6	0	95.8	0
	0.4	1800.0	22.4%	244.9	0	168.6	0	160.4	0	169.9	0	119.1	0
(23, 8, 7)	0.05	1800.0	22.9%	1261.8	7.6%	1127.8	4.1%	1105.2	3.4%	1107.2	4.2%	993.4	3.2%
	0.2	1800.0	21.1%	1259.8	6.7%	1030.1	4.5%	1051.5	4.9%	1031.1	5.2%	902.1	4.9%
	0.4	1800.0	21.2%	1101.4	7.2%	925.7	9.6%	929.2	9.5%	942.0	9.6%	772.8	7.2%
(25, 9, 8)	0.05	1800.0	24.2%	1514.1	6.6%	1443.1	5.3%	1468.6	6.0%	1478.2	5.9%	1156.5	4.2%
	0.2	1800.0	19.1%	1643.1	7.3%	1596.3	8.6%	1589.8	5.5%	1588.7	6.6%	1472.8	8.0%
	0.4	1800.0	22.8%	1753.2	9.8%	1719.0	7.6%	1742.6	7.1%	1742.8	9.4%	1680.8	5.5%

Table 4-6 indicates that CPA3 outperforms all other variants. In particular, CPA3 outperforms CPA1 as predicted by Theorem 4.3. In fact, CPA1 failed to terminate within 1800 seconds on any instance having $|V| = 23$ or 25 nodes. Note that CPA3 is faster than all of the CPA2 variants, which employ valid inequalities 4–12. Moreover,

the average optimality gap produced by CPA3 is not worse than that produced by CPA2 variants in general. This may stem from the fact that the problem DEF-R in CPA2 variants is augmented with extra variables (due to the presence of x^L-variables), resulting in a mixed-integer problem that is harder to solve. Furthermore, recall that valid inequalities 4–12 are not necessarily stronger than 4–13 due to the fact that we reduce some coefficients of x_i-variables in 4–13 that cannot be reduced in 4–12. This observation, along with the modest tightening step afforded by Algorithm 1, also explains why CPA3 outperforms the CPA2 variations. Also, note that CPA2-1 is outperformed by other CPA2 variants, although the difference between CPA2-2, CPA2-3, and CPA2-4 is insignificant especially for smaller instances.

CHAPTER 5
CONCLUSIONS AND FUTURE RESEARCH

In Chapter 2, we addressed a finite-horizon optimal stopping problem from the seller's perspective. We began by demonstrating that when the customer is optimal, the seller can optimize profit from selling items in $O(n \log n)$ time, where n is the number of items for sale. The vast literature in experimental research on stopping problems has shown that human decision-makers, acting as the customer, tend to stop search too soon, and in any case cannot be assumed to be optimal decision-makers. We modeled the unpredictability of human decision-making behavior by analyzing situations in which the items' values, profits, and customer stopping thresholds are uncertain. We first examined a max-min case in which the seller wishes to maximize the minimum profit that can be made given some uncertainty set in which the data values must reside. A special case of this max-min problem that we studied in Chapter 2 remains polynomially solvable. Next, we examined the case in which the seller wishes to maximize expected profit. This problem turns out to be NP-hard, even when uncertainty is confined to the items' values.

We provided a formulation for solving the problem of maximizing expected profit (in which uncertainty can be applied to any part of the data except for n). However, we did not explore solution techniques tailored for this problem, beyond the use of standard mixed-integer programming solvers. When n or $|Q|$ is large, it is not likely that formulation (2–11) will be tractable. One area of future research may instead focus on custom solution techniques for solving (2–11) within reasonable computational limits. Another area of interest is certainly in laboratory testing of these models. Conservative models (such as those presented in Section 2.3) tend to sacrifice potential profit in favor or guaranteeing minimum profits. It would be of interest to demonstrate how conservative the seller should be in practice given a human decision-maker. Furthermore, we have assumed that the items' values and profits are independent in

general. As an extension to our work, the scenarios in which there exist a degree of correlation between values and profits can also be considered for future studies. Finally, an expanded version of this problem may attempt to observe this game in a repeated setting, in which the customer adapts the purchasing strategy based on the tendencies of a profit-motivated seller.

In Chapter 3, we studied a version of the set covering problem in which items are used to cover clauses, and where each clause has a prioritized list on which items would be used to cover the clause. The clause is then satisfied by the selected item having the highest priority. We considered a two-player Stackelberg game in which players introduce items in turn, and then earn a reward for each clause that they satisfy. The key assumptions are that the follower acts with knowledge of the leader's decision, and that the follower acts to maximize its own objective (rather than, e.g., minimizing the leader's objective). We formulated a mixed-integer bilevel programming model for the problem, along with a cutting-plane algorithm for solving the problem. We showed that our family of approaches is computationally preferable to general bilevel optimization approaches that have been previously developed.

For future research, there are many implementation challenges that can be investigated under this approach. The augmented CPA relies on an approach that restricts the possible follower actions in (3–10), and as such relaxes the outer optimization problem in that formulation. In our computational experiments, the ACPA implementations occasionally show some promise but are inconsistent in successfully verifying the validity of the candidate inequalities. Therefore, it would appear that there exists an opportunity to investigate tighter relaxations of (3–10), which would allow the validation process to more accurately assess whether or not a candidate inequality is valid, thus resulting in faster implementations. Another line of research might seek to derive locally valid inequalities on "rational" follower reactions within the branch-and-bound tree, based on branching decisions for the leader's decisions at a particular node of the tree. Finally,

hybridizing branch-and-bound search with heuristic strategies for the follower may prove useful in a near-optimal algorithmic approach, especially for those instances that appear to resist exact solution methods.

In Chapter 4, we addressed a Stackelberg game on a network in which an attacker seeks to spread influence on the nodes over a finite number of time stages. The defender thus aims to protect the network against the spread of the attacker's influence. By devising several valid inequalities discussed in Section 4.3, we proposed an exact cutting-plane algorithm that is capable of finitely identifying an optimal solution for the defender's problem. We also developed alternative formulations for the attacker's problem in Section 4.4, and studied different characteristics of each formulation. In particular, we proposed a solution method for the attacker's problem based on Benders' decomposition in which the cuts are calculated using a polynomial-time cut generating scheme that does not require solving a linear programming subproblem.

Several extensions can be considered for the research work presented in Chapter 4. First, recall that several spread networks may correspond to one optimal solution for the attacker's problem. Our approach to generating spread network inequalities starts with some spread network, and modifies it to generate a stronger valid inequality. As a result, the strength of the identified valid inequality is dependent on the initially-chosen spread network. Thus, a future task might focus on optimizing the structure of a spread network in order to produce a strongest possible valid inequality.

A second area of research may consider other diffusion models aside from the threshold model examined in Chapter 4. For instance, it is interesting to extend our model to cases in which the neighbors of each node may have unequal effects on the node. (For example, the influence of node v over another node w may be represented by some parameter b_{vw}, and node w may become influenced if $\sum_{v \in V} b_{vw}$ exceeds some given threshold. See [42] for further details.) Another line of research is to incorporate uncertainty in the diffusion process. One such process discussed by Goldenberg et al.

[38] addresses the situation in which once a node is influenced, it is given one chance to influence its neighbors according to a Bernoulli distribution.

A third line of research may investigate a different Stackelberg game in which each player aims to spread its own influence by seeking a subset of nodes to initially attack. In this game, the objective of each player is to maximize the reward obtained from influencing nodes with respect to some budget restriction. These settings often lead to bilevel programs, with the follower's optimization problem embedded in the constraints of the leader's problem. For such problems, ideas similar to the reformulation given in Section 4.3.1 for the defender's problem may be promising in devising exact cutting-plane solution methods.

APPENDIX A
APPENDIX ON REPRESENTATION OF THRESHOLD VALUES

In order to precisely discuss the complexity of the problems under investigation here, we must address the size of the data used in our computations. Even after making the simplifying assumption that the customer's values are uniformly distributed on the interval [0,100], it is not clear that the customer can truly solve the optimal purchasing (stopping) problem in polynomial time. The recursions in (2–1a) and (2–1b) allow the generation of threshold data in $O(n)$ time provided that computations are performed in constant time. However, note that (after dividing the maximum customer values by 100) $t_n = 1/2$, and that $t_i = (t_{i+1}^2 + 1)/2$ for each $i = 1, \ldots, n-1$. This means that $t_{n-1} = 5/8$, $t_{n-2} = 89/128$, and so on: The implication is that $t_{n-j+1} = \alpha_j/(2^{2^j-1})$ for some integer numerator α_j, $\forall j = 1, \ldots, n$. Unfortunately, this implies that the number of bits required to store α-values is $O(2^n)$. Therefore, it is not technically permissible to let $t^\star = (t_{n+1} + t_{n+2})/2$ in the proof of Theorem 2.2, because storing this value evidently requires an exponential number of bits. (In fact, it is more accurate to say that we do not know how to store this number using a polynomial number of bits.)

A simplifying assumption would state that the customer makes all computations with finite precision, and that this level of precision is treated as a constant value in our computational analysis. But interestingly, for the case in which the customer perceives a uniform distribution of probability data, Theorem 2.2 holds true even when no assumption is made that restricts the precision of the customer's computations. We discuss the details of this argument below.

Consider the customer's optimal stopping problem with n total items. We seek a sequence of values $s_1, \ldots, s_n$ such that $t_n = 0.5 \leq s_n < t_{n-1} \leq s_{n-1} < \cdots < t_1 \leq s_1$, where $s_{n+1-j} = \beta_j/2^{j+2}$ for $j = 1, \ldots, n$, such that each β_j is a positive integer and can thus be represented using no more than $n+2$ bits. It is easy to show that $\beta_1 = 4$, $\beta_2 = 10$, $\beta_3 = 23$, and $\beta_4 = 48$ are valid in the sense that they generate s-values that satisfy the

inequality chain above, after dividing by 8, 16, 32, and 64, respectively. For $j = 5$, we can select β_5 to be any value in $\{100, 101, 102\}$. Using $j = 5$ as a base case, we will prove by induction that for any $j \geq 5$, there exist at least three values of β_j such that $s_{n+1-j} = \beta_j/2^{j+2}$ is valid.

Suppose that this property holds for a given $j \geq 5$. We thus have:

$$t_{n+1-j} \leq \beta_j/2^{j+2} < (\beta_j + 2)/2^{j+2} < t_{n-j},$$

for some β_j. Consider now the computation of a threshold value β_{j+1} via the recursion (2–1b). Note that a threshold τ'_{j+1} based on the value $\beta_j/2^{j+2}$ would be computed as $(\beta_j^2 + 2^{2(j+2)})/2^{2(j+2)+1}$, and a threshold τ''_{j+1} based on the value $(\beta_j + 2)/2^{j+2}$ would be computed as $(\beta_j^2 + 4\beta_j + 4 + 2^{2(j+2)})/2^{2(j+2)+1}$. However, to use these thresholds for s-values, the denominator must be no more than 2^{j+3} rather than $2^{2(j+2)+1}$. Hence, consider the following threshold values, obtained after dividing the numerator and denominator by 2^{j+2} and rounding the numerator so that it is integer-valued:

$$\tau'_{j+1} = \frac{\beta'_{j+1}}{2^{j+3}} = \frac{\lceil (\beta_j^2 + 2^{2(j+2)})/2^{j+2} \rceil}{2^{j+3}} \tag{A–1}$$

$$\tau''_{j+1} = \frac{\beta''_{j+1}}{2^{j+3}} = \frac{\lfloor (\beta_j^2 + 4\beta_j + 4 + 2^{2(j+2)})/2^{j+2} \rfloor}{2^{j+3}}. \tag{A–2}$$

If $\beta'_{j+1} \leq \beta''_{j+1}$, then β_{j+1} can validly take on any integer value in the interval $[\beta'_{j+1}, \beta''_{j+1}]$. We show here that $\beta'_{j+1} + 2 \leq \beta''_{j+1}$. Note that when $j = 5$, we have that $t_{n-4} > 3/4$. Comparing the difference in the numerators in (A–2) and (A–1) before rounding, we have:

$$\frac{\left(\beta_j^2 + 4\beta_j + 4 + 2^{2(j+2)}\right) - \left(\beta_j^2 + 2^{2(j+2)}\right)}{2^{j+2}} = \frac{4\beta_j + 4}{2^{j+2}} > 3,$$

where the latter inequality is due to the fact that $\beta_j/2^{j+2} \geq t_{n-4} > 3/4$. Performing the ceiling and floor operations on the numerators of (A–1) and (A–2), respectively, narrows

the gap between these values to at least two, which verifies our claim. (Observe now that we have required $\beta'_j + 2 \leq \beta''_j$ so that we are guaranteed to have a nonempty interval $[\beta'_{j+1}, \beta''_{j+1}]$ when using the above induction argument.)

Therefore, we can compute the β-values as given by the base cases above for $j = 1, \ldots, 5$, and then by recursion using (A–1) thereafter, using a polynomial number of bits. Hence, in the uniform distribution case, Theorem 2.2 is still valid even when the customer uses infinite precision, when we select $t^\star = s_{n+1}$ in that transformation, assuming that $j \geq 5$ (with the case of $j \leq 4$ being trivial). This guarantees that $t_{n+1} > t^\star > t_{n+2}$, and that $t^\star$ is encodable using a polynomial number of bits.

APPENDIX B
PROOF OF THEOREM 3.1

We prove that $F_{\bar{x}}$ is NP-hard in the strong sense by providing a polynomial reduction from EXACT COVER BY THREE SETS (X3C) [35] to a decision version of $F_{\bar{x}}$. X3C is defined with a set of elements $S = \{1, \dots, 3p\}$, and a collection of $q > p$ subsets of S, $\mathcal{A} = \{S_1, \dots, S_q\}$, each having a cardinality of three.

To transform X3C to a decision version of $F_{\bar{x}}$, we let $M = \{1, \dots, 3p\}$ and $N = \{1, \dots, q\}$. The follower incurs an introduction cost of 1 for each product. Each preference list, O_i, contains all products $j : i \in S_j$, which are ordered arbitrarily. Also, let $r_{ij} = \rho_{ij} = 1, \ \forall i \in M, \ j \in N$. The leader chooses not to introduce any products. Then a decision version of the follower problem is as follows: does there exist a set of products that yields a profit of $2p$ for the follower? We show that there exists an exact cover of S by a subset of $\mathcal{A}$ if and only if the follower's maximum profit is $2p$.

First we prove if there exist a solution to X3C, the follower can make a profit of $2p$. Suppose $\bar{\mathcal{A}} \subset \mathcal{A}$ is an exact cover, and that the follower introduces products $j : S_j \in \bar{\mathcal{A}}$. For every $i \in M$, note that because $\bar{\mathcal{A}}$ is an exact cover of $\mathcal{A}$, exactly one product j, for some $j : i \in S_j$, has been introduced. Hence the follower earns a revenue of 1 from all $3p$ customers. Because $|\bar{\mathcal{A}}| = p$, the follower spent p in introducing products and earned a profit of $2p$.

Next, suppose that the follower can make a profit of $2p$. We show that the set of introduced products corresponds to a solution for X3C. The follower must introduce exactly p products to obtain a profit of $2p$. Introducing $p' < p$ products yields a maximum revenue of $3p'$, and a cost of p' for a profit of $2p' < 2p$. On the other hand introducing $p' > p$ products incurs a cost of p', with a maximum revenue of $3p$; the profit would be no more than $3p - p' < 2p$. If exactly p products are introduced, a profit of $2p$ is obtained if and only if a revenue of $3p$ is achievable, i.e., every product is purchased by three

customers. Define

$$\bar{\mathcal{A}} = \{S_j \in \mathcal{A} : \text{ product } j \text{ is introduced}\}.$$

Since every introduced product was purchased by three customers, we must have $S_{k_1} \cap S_{k_2} = \emptyset, \ \forall k_1, k_2 : \ S_{k_1}, S_{k_2} \in \bar{\mathcal{A}}$. Hence $\bar{\mathcal{A}}$ solves X3C.

Because all numerical data used in this transformation equals 1, and because the number of customers and products is polynomially bounded by the X3C problem size, we conclude that the decision version of the follower problem is strongly NP-complete, and that the follower's optimization problem is strongly NP-hard.

REFERENCES

[1] Audet, C., Savard, G., and Zghal, W. "New Branch-and-Cut Algorithm for Bilevel Linear Programming." *Journal of Optimization Theory and Applications* 134 (2007): 353–370.

[2] Barabási, A.-L. and Albert, R. "Emergence of scaling in random networks." *Science* 286 (1999): 509–512.

[3] Bard, J. F. *Practical Bilevel Optimization: Algorithms and Applications*. Boston: Kluwer Academic Publishers, 1998.

[4] Bartoszynski, R. and Govindarajulu, Z. "The secretary problem with interview cost." *Sankhyā: The Indian Journal of Statistics* 40 (1978): 11–28.

[5] Bayus, B. L., Jain, S., and Rao, A. G. "Truth or Consequences: An Analysis of Vaporware and New Product Announcements." *Journal of Market Research* 38 (2001): 3–13.

[6] Bearden, J. N. and Rapoport, A. "Operations research in experimental psychology." *Tutorials in Operations Research: Emerging Theory, Methods, and Applications*. ed. J. C. Smith, vol. 1. INFORMS, Linthicum, MD, 2005. 213–236.

[7] Bearden, J. N., Rapoport, A., and Murphy, R. O. "Sequential observation and selection with rank-dependent payoffs: An experimental test." *Management Science* 52 (2006): 1437–1449.

[8] Bellman, R. *Dynamic Programming*. Princeton, NJ: Princeton University Press, 1957.

[9] Ben-Tal, A., El Ghaoui, L., and Nemirovski, A. *Robust Optimization*. Princeton Series in Applied Mathematics. Princeton University Press, Princeton, NJ, 2009.

[10] Bialas, W. F. and Karwan, M. H. "Two-Level Linear Programming." *Management Science* 30 (1984).8: 1004–1020.

[11] Birge, J. R. and Louveaux, F. V. *Introduction to Stochastic Programming*. New York: Springer, 1997.

[12] Blum, J., Ding, M., Thaeler, A., and Cheng, X. "Connected Dominating Set in Sensor Networks and MANETs." *Handbook of Combinatorial Optimization Supplement Volume B*. eds. D. Du and P. M. Pardalos. Springer, 2005. 329–369.

[13] Brown, G. G., Carlyle, W. M., Salmerón, J., and Wood, K. "Defending Critical Infrastructure." *Interfaces* 36 (2006).6: 530–544.

[14] Brown, G. G., Carlyle, W. M., Salmerón, J., and Wood, R. K. "Analyzing the Vulnerability of Critical Infrastructure to Attack and Planning Defenses." *Tutorials in Operations Research: Emerging Theory, Methods, and Applications*. eds. Harvey J. Greenberg and J. Cole Smith. Hanover, MD: INFORMS, 2005. 102–123.

[15] Budak, C., Agrawal, D., and El Abbadi, A. "Limiting the spread of misinformation in social networks." *Proceedings of the Twentieth International Conference on World Wide Web*. New York, NY: ACM, 2011.

[16] Calvete, H. I. and Galé, C. "A note on Bilevel Linear Fractional Programming problem." *European Journal of Operational Research* 152 (2004).1: 296–299.

[17] Candler, W. and Norton, R. "Multilevel programming." Tech. Rep. 20, World Bank Development Research Center, Washington, DC, 1977.

[18] Candler, W. and Townsley, R. "A linear two-level programming problem." *Computers and Operations Research* 9 (1982).1: 59–76.

[19] Chen, N. "On the Approximability of Influence in Social Networks." *SIAM Journal of Discrete Mathematics* 23 (2009).3: 1400–1415.

[20] Chow, Y. S., Robbins, H., and Siegmund, D. *Great expectations: the theory of optimal stopping*. Houghton Mifflin, 1971.

[21] Chung, F. and Lu, L. *Complex graphs and networks*. Providence, RI: American Mathematical Society, 2006.

[22] Church, R. L. and Scaparra, M. P. "The r-interdiction median problem with fortification." *Geographical Analysis* 39 (2007): 129–146.

[23] Church, R. L., Scaparra, M. P., and Middleton, R. S. "Identifying critical infrastructure: The median and covering facility interdiction problems." *Annals of the Association of American Geographers* 94 (2004): 491–502.

[24] Colson, B., Marcotte, P., and Savard, G. "An overview of bilevel optimization." *Annals of Operations Research* 153 (2007): 235–256.

[25] Dempe, S. *Foundations of Bilevel Programming*. Boston: Kluwer Academic Publishers, 2002.

[26] Dempe, S. "Annotated Bibliography on Bilevel Programming and Mathematical Programs with Equilibrium Constraints." *Optimization* 52 (2003).3: 333–359.

[27] DeNegre, S. and Ralphs, T. K. "A Branch-and-Cut Algorithm for Bilevel Integer Programming." *Proceedings of the Eleventh INFORMS Computing Society Meeting*. Charleston, SC, pages 65-78, 2009.

[28] Dinh, T. N., Nguyen, D. T., and Thai, M. T. "Cheap, easy, and massively effective viral marketing in social networks: truth or fiction?" *Proceedings of the Twenty Third ACM Conference on Hypertext and Social Media*. New York: ACM, 2012, 165–174.

[29] Faisca, N. P., Dua, V., Rustem, B., Saraiva, P. M., and Pistikopoulos, E. "Parametric global optimisation for bilevel programming." *Journal of Global Optimization* 38 (2007): 609–623.

[30] Farrell, J. and Saloner, G. "Installed Base and Compatibility: Innovation, Product Preannouncements and Predation." *American Economic Review* 76(5) (1986): 940–955.

[31] Ferguson, T. S. "Who solved the secretary problem?" *Statistical Science* 4 (1989): 282–296.

[32] Ferguson, T. S. "Optimal Stopping and Applications." http://www.math.ucla.edu/~tom/Stopping/Contents.html, 2010.

[33] Freeman, P. R. "The secretary problem and its extensions: A review." *International Statistical Review* 51 (1983): 189–206.

[34] Gardner, M. "Mathematical games." *Scientific American* 202 (1960): 150–153.

[35] Garey, M. R. and Johnson, D. S. *Computers and Intractability: A Guide to the Theory of NP-completeness*. Princeton, NJ: W. H. Freeman & Co., 1979.

[36] Gendreau, M., Marcotte, P., and Savard, G. "A hybrid Tabu-ascent algorithm for the linear Bilevel Programming Problem." *Journal of Global Optimization* 8 (1996): 217–233.

[37] Gilbert, J. and Mosteller, F. "Recognizing the maximum of a sequence." *Journal of the American Statistical Association* 61 (1966): 35–73.

[38] Goldenberg, J., Libai, B., and Muller, E. "Talk of the Network: A Complex Systems Look at the Underlying Process of Word-of-Mouth." *Marketing Letters* 12 (2001): 211–223.

[39] Guo, J., Niedermeier, R., and Raible, D. "Improved Algorithms and Complexity results for Power Domination in Graphs." *Fundamentals of Computation Theory*. eds. Maciej Liskiewicz and Rdiger Reischuk, vol. 3623 of *Lecture Notes in Computer Science*. Berlin: Springer, 2005. 172–184.

[40] Hammer, P.L. and Rudeanu, S. *Boolean Methods in Operations Research and Related Areas*. Berlin: Springer, 1968.

[41] Hejazi, S.R., Memariani, A., Jahanshahloo, G., and Sepehri, M.M. "Linear bilevel programming solution by genetic algorithm." *Computers and Operations Research* 29 (2002).13: 1913–1925.

[42] Kempe, D., Kleinberg, J., and Tardos, E. "Maximizing the Spread of Influence Through a Social Network." *Proceedings of the Ninth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. New York: ACM, 2003, 137–146.

[43] Lee, M. D. "A hierarchical Bayesian model of human decision-making on an optimal stopping problem." *Cognitive Science* 30 (3) (2006): 555–580.

[44] Leskovec, J., Krause, A., Guestrin, C., Faloutsos, C., VanBriesen, J., and Glance, N. "Cost-effective outbreak detection in networks." *Proceedings of the Thirteenth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 2007, 420–429.

[45] Liberatore, F., Scaparra, M. P., and Daskin, M. S. "Analysis of facility protection strategies against an uncertain number of attacks: The stochastic R-interdiction median problem with fortification." *Computers and Operations Research* 38 (2010).1: 357–366.

[46] Lilly, B. and Walters, R. "Toward a Model of New Product Preannouncement Timing." *Journal of Product Innovation Management* 14 (1997): 4–20.

[47] Migdalas, A., Pardalos, P. M., and Varbrand, P. *Multilevel Optimization*. Boston: Kluwer Academic Publishers, 1998.

[48] Mishra, D. P. and Bhabra, H. S. "Assesing the economic worth of new product pre-announcement signals: Theory and empirical evidence." *Journal of Product and Brand Management* 10(2) (2001): 75–93.

[49] Mitsos, A. "Global solution of nonlinear mixed-integer bilevel programs." *Journal of Global Optimization* 47 (2010): 557–582.

[50] Mitsos, A., Lemonidis, P., and Barton, P. "Global solution of bilevel programs with a nonconvex inner program." *Journal of Global Optimization* 42 (2008): 475–513.

[51] Moore, J. T. and Bard, J. F. "The Mixed Integer Linear Bilevel Programming Problem." *Operations Research* 38 (1990).5: 911–921.

[52] Nemhauser, G. L., Wolsey, L. A., and Fisher, M. L. "An Analysis of the Approximations for Maximizing Submodular Set Functions." *Mathematical Programming* 14 (1978): 265–294.

[53] Nemhauser, George L. and Wolsey, Laurence A. *Integer and Combinatorial Optimization*. New York, NY: Wiley-Interscience, 1999.

[54] Nguyen, N. P., Yan, G., Thai, M. T., and Eidenbenz, S. "Containment of misinformation spread in online social networks." *Proceedings of the Fourth ACM Web Science*. 2012.

[55] Nicol, D. and Liljenstam, M. "Models and Analysis of Active Worm Defense." *Proceedings of the Third International Workshop on Mathematical Models, Architectures and Protocols for Computer Network Security (MMM-ACNS)*. 2005, 38–53.

[56] Ozaltin, O. Y., Prokopyev, O. A., and Schaefer, A. J. "The bilevel knapsack problem with stochastic right-hand sides." *Operations Research Letters* 38 (2010).4: 328–333.

[57] Pressman, E. L. and Sonin, I. M. "The best choice problem for a random number of objects." *Theory of Probability and Its Applications* 17 (1972): 657–668.

[58] Ramirez-Marquez, J. E., R., C., Pohl, E., and Medal, H. "Facility Location with Interdiction: A Multi-Objective Analysis." Tech. rep., School of Systems and Enterprises, Stevens Institute of Technology, Hoboken, New Jersey, 2012.

[59] Samuels, S. M. "Secretary Problems." *Handbook of Sequential Analysis*. eds. B. K. Gosh and P. K. Sen. Marcel Dekker, New York, 1991. 381–405.

[60] Scaparra, M. P. and Church, R. L. "A bilevel mixed-integer program for critical infrastructure protection planning." *Computers and Operations Research* 35 (2008).6: 1905–1923.

[61] Seale, D. A. and Rapoport, A. "Sequential decision making with relative ranks: An experimental investigation of the secretary problem." *Organizational Behavior and Human Decision Processes* 69 (1997): 221–236.

[62] Shen, S. "Domination Problems." *Encyclopedia of Operations Research and Management Science*. ed. J. J. Cochran. Hoboken, NJ: Wiley, 2010.

[63] Shen, S., Smith, J. C., and Goli, R. "Exact Interdiction Models and Algorithms for Disconnecting Networks via Node Deletions." *Discrete Optimization* 9 (2012).3: 172–188.

[64] Shen, Y., Dinh, T. N., Zhang, H., and Thai, M. T. "Interest-Matching Information Propagation in Online Social Networks." *ACM International Conference on Information and Knowledge Management (CIKM)*. 2012.

[65] Shimizu, K. and Aiyoshi, E. "A new computational method for Stackelberg and min-max problems by use of penalty method." *IEEE Transactions on Automatic Control* 26 (1981): 460–466.

[66] Smith, J. C. "Basic Interdiction Models." *Wiley Encyclopedia of Operations Research and Management Science*. ed. J. Cochran. Hoboken, NJ: Wiley, 2010. 323–330.

[67] Smith, J. C. and Lim, C. "Algorithms for Network Interdiction and Fortification Games." *Pareto Optimality, Game Theory and Equilibria*. eds. A. Migdalas, P. M. Pardalos, L. Pitsoulis, and A. Chinchuluun, Nonconvex Optimization and its Applications Series. New York: Springer, 2008. 609–644.

[68] Smith, J. C., Lim, C., and Alptekinoglu, A. "New product introduction against a predator: A bilevel mixed-integer programming approach." *Naval Research Logistics* 56 (2009).8: 714–729.

[69] Smith, M. H. "A secretary problem with uncertain employment." *Journal of Applied Probability* 12 (1975): 620–624.

[70] Tanachaiwiwat, S. and Helmy, A. "Encounter-based worms: Analysis and defense." *Ad Hoc Network* 7 (2009).7: 1414–1430.

[71] Thirwani, D. and Arora, S. R. "An algorithm for the integer linear fractional bilevel programming problem." *Optimization* 39 (1997).1: 53–67.

[72] Vicente, L. N. and Calamai, P. H. "Bilevel and multilevel programming: A bibliography review." *Journal of Global Optimization* 5 (1994): 291–306.

[73] von Stackelberg, H. *The Theory of the Market Economy*. London, U.K.: William Hodge and Co., 1952.

[74] Wen, U. P. and Hsu, S. T. "Linear Bi-Level Programming Problems – A Review." *The Journal of the Operational Research Society* 42 (1991).2: 125–133.

[75] Wood, R. K. "Deterministic network interdiction." *Mathematical and Computer Modelling* 17 (1993).2: 1–18.

[76] Wu, J., Cardei, M., Dai, F., and Yang, S. "Extended Dominating Set and Its Applications in Ad Hoc Networks Using Cooperative Communication." *IEEE Transactions on Parallel and Distributed Systems* 17 (2006).8: 851–864.

[77] Zhao, M., Kang, L., and Chang, G. J. "Power domination in graphs." *Discrete Mathematics* 306 (2006).15: 1812–1816.

BIOGRAPHICAL SKETCH

Mehdi Hemmati (Soheil) was born in Tehran, Iran. He graduated from Alborz High School at Tehran in 1998 and decided to pursue an engineering degree in college. He was admitted to Sharif University of Technology in fall 1998 and received his bachelor's and master's degrees in industrial and systems engineering from the same university in 2003 and 2005, respectively. In fall 2009, he received alumni graduate award to study for Ph.D. degree in the Department of Industrial and Systems Engineering at the University of Florida. His research centered on multilevel discrete optimization and interdiction theory with applications that involve competition, either between two agencies (e.g., in market) or against uncertain external factors. He received his Ph.D. degree in August 2013.