Stochastic Combinatorial Optimization with Applications in Graph Covering

Hao-Hsiang Wu

A dissertation
submitted in partial fulfillment of the
requirements for the degree of

Doctor of Philosophy

University of Washington

2018

Reading Committee:

Simge Küçükyavuz, Chair

Archis Ghate

Shan Liu

Program Authorized to Offer Degree:
Industrial and Systems Engineering

University of Washington

Abstract

Stochastic Combinatorial Optimization with Applications in Graph Covering

Hao-Hsiang Wu

Chair of the Supervisory Committee:
Associate Professor Simge Küçükyavuz
Industrial and Systems Engineering

We study stochastic combinatorial optimization models and propose methods for their solution. First, we consider a risk-neutral two-stage stochastic programming model for which the objective value function of the second-stage subproblems is submodular. Next, we consider risk-averse combinatorial optimization problems, where in one variant, the risk is measured with a chance constraint, and in another variant, conditional value-at-risk is used to quantify risk. We demonstrate the proposed models and methods on various graph covering problems. We provide our research scope and a review of fundamental models in Chapter 1.

In Chapter 2, we introduce a new class of problems that we refer to as two-stage stochastic submodular optimization models. We propose a delayed constraint generation algorithm to find the optimal solution to this class of problems with a finite number of samples. We apply the generic model and method to stochastic influence maximization problems arising in social networks. Consider a covering problem on a random graph, where there is uncertainty on whether an arc appears in the graph. The problem aims to find a subset of nodes that reaches the largest expected number of nodes in the graph. In contrast to existing studies that involve greedy approximation algorithms with a 63% performance guarantee, our work focuses on solving the problem optimally. We show that the submodularity of the influence function can be exploited to develop strong optimality cuts that are more effective than the standard optimality cuts available in the literature. We report our computational

experiments with large-scale real-world datasets for two fundamental influence maximization problems, independent cascade and linear threshold, and show that our proposed algorithm outperforms the basic greedy algorithm of Kempe et al. (2003).

In Chapter 3, we investigate a class of chance-constrained combinatorial optimization problems. The chance-constrained program aims to find the minimum cost selection of a vector of binary decisions such that a desirable event occurs with a high probability. For a given decision, we assume that we have an oracle that computes the probability of a desirable event exactly. Using this oracle, we propose an exact general method for solving the chance-constrained problem. Furthermore, we show that if the chance-constrained program is solved approximately by a sampling-based approach, then the oracle can be used as a tool for checking and fixing the feasibility of the optimal solution given by this approach. We demonstrate the effectiveness of our proposed methods on a probabilistic partial set covering problem (PPSC). We give a compact mixed-integer program that solves PPSC optimally (without sampling) for a special case. For large-scale instances for which the exact methods exhibit slow convergence, we propose a sampling-based approach that exploits the submodular structure of PPSC. In particular, we introduce a new class of facet-defining inequalities for a submodular substructure of PPSC and show that a sampling-based algorithm coupled with the probability oracle solves the large-scale test instances effectively.

In Chapter 4, we study a class of risk-averse submodular maximization problems that optimizes the conditional value-at-risk (CVaR) of a random objective function at a given risk level, where the random objective function is defined as a nondecreasing submodular set function. We assume that we have an oracle that computes the CVaR of the random objective function exactly. Using this oracle, we propose an exact general method for solving this problem. Furthermore, we show that the problem can be solved approximately by a sampling-based approach. We demonstrate the proposed methods on a variant of stochastic set covering problem.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACKNOWLEDGMENTS

I would like to start by thanking my advisor, Dr. Simge Küçükyavuz. Without her expertise, patience, and encouragement, I would not have been able to complete this dissertation within these four years. I also appreciate the Graduate Research Assistantship and travel support from her (National Science Foundation grants #1732364 and #1733001). She is a positive energizer, who instructs me on how to be an operations research scientist. Dr. Küçükyavuz goes above and beyond in supporting my research. I respect her leadership, wisdom, dedication, and responsibility. Dr. Küçükyavuz is my role model, leading by example and demonstrating the open mind of wisdom. I sincerely appreciate not only the opportunities but the genuine advice she has given me throughout these years. Without a doubt, Dr. Simge Küçükyavuz has been one of the most influential persons in my life.

Moreover, I am particularly grateful to Dr. Archis Ghate and Dr. Shan Liu for serving on my reading committee and providing valuable feedback. Special thanks to the graduate school representative, Dr. Thomas Rothvoß, for providing me with constructive suggestions in the general and final exams. Thank you also to our department BSIE/Ph.D. adviser, Ms. Jennifer Tsai, for supporting me whenever I needed help. I would like to thank Dr. Marc Posner for offering Graduate Teaching Assistantship (Courses: Linear Optimization and Operations Research Models and Methods) at the Ohio State University. It was a delightful moment that I spent time on teaching students or learning a great deal about teaching from Dr. Posner. I have had numerous opportunities to learn from outstanding teachers in graduate studies. I would like to specifically thank Dr. Yuh-Dauh Lyuu, Dr. Mi-Yen Yeh, Dr. Pei-Yin Chen, Dr. Chih-Ping Chu, Dr. Si-Kuen Lee, Dr. Sy-Shyan Chen, and Dr. Patty Buchanan for inspiring me to become a better version of myself.

I would like to thank some excellent graduate students, Jinyuan Zhang, Xuewen Jiang, Rui Xue, James Li, Jie Gao, Evan Mirolla, and Minzheng Chen, for sharing their areas of expertise and being delightful friends. Special thanks to Yanzhuang Zhang and Kuang-Shun Yang for sharing their Ph.D. journeys and positive attitudes with me. Thanks to Chu-Yun Lin and Kyle Stuart Chang for more than fifteen years friendship. Thanks to Ya-Yun Cheng and Ying-Chun Chen for listening to me and sharing their life experiences with me. Thanks to Ching-Hsiang Chu for helping me a lot when I started out as a graduate student at the Ohio State University. I would like to thank the members of the Taiwanese Basketball Club for continuing my basketball life at the Ohio State University. I still remember those moments when we were training together in RPAC and had many beautiful games in different tournaments. Special thanks to Mo-Sheng Chiu, Tseh Jing, Zeming Yin, and Anqi Wei for their aggressive offense and solid defense in the paint. I would like to thank my officemates including Cheng-Lung Chen, Xiao Liu, Benjamin Chaiken, Aven Samareh, and Hasan Manzour for being my accommodating members. Special thanks to Ting-Yu Ho and his lovely son for supporting me and warming my heart. From Ohio to Washington, I was fortunate to have had a clever, kindhearted, and warm colleague, Merve Meraklı, in the research group. I am grateful to Merve for listening to and encouraging me when I struggled with my research.

Last but not least, I would like to thank my mother and sister, Tai-Chia Liu and Hao-Yi Wu, for their infinite capacity for love. I would like to express my special thanks to Ko-Hsin Chang for accompanying me in the darkest night. I am also indebted to others who have treated me like a family member or a trustworthy friend. If you are looking for your name here, I also thank you and profoundly apologize for my carelessness. Finally, I wish to express my gratitude to my toughest father, Yu-Ming Wu (1954-2013), who always encourages me in my mind.

DEDICATION

to my parents, Tai-Chia Liu and Yu-Ming Wu

VITA

Education	**University of Washington**	**Seattle, WA**
	Ph.D., Industrial and Systems Engineering	2018
	The Ohio State University	**Columbus, OH**
	M.S., Operations Research	2016
	National Taiwan University	**Taipei, Taiwan**
	M.S., Computer Science and Information Engineering	2013
	National Cheng Kung University	**Tainan, Taiwan**
	B.S., Computer Science and Information Engineering	2010
Experience	**Taiwan Armed Forces**	**Hsinchu, Taiwan**
	Reserve Officer (Second Lieutenant)	Aug 2013 - July 2014

Chapter 1
INTRODUCTION

In this dissertation, we consider *stochastic combinatorial* optimization problems. In this chapter, we give a brief review of the fundamental models and methodologies. At the end of this chapter, we introduce the research scope and the dissertation outline.

1.1 Stochastic Combinatorial Optimization

Combinatorial optimization is a discrete mathematical optimization problem that aims to find an optimal subset from a finite set of components. More formally, let V be a finite set, $\mathcal{F}$ be a set of feasible subsets of V, and g be an objective function $g : 2^V \to \mathbb{R}$. A combinatorial optimization problem (COP) has a general form

$$\min_{S \in \mathcal{F}} g(S). \tag{1.1}$$

Formulation (1.1) is the minimization form for COPs. The maximization form can be derived by substituting "min" with "max". COPs widely exist in operations research, computer science, and many other areas. For example, in operations research, the traveling salesman problem and the set covering problem have applications for supply chain management and logistics. In computer science, the minimal spanning tree problem is applicable to computer communication networks, and the influence maximization of [38] arises in the social networks. We refer the reader to [66, 74] for a review of the various models and applications of COPs. Although in some special cases COPs can be solved in polynomial time, most of them are proven to be $\mathcal{NP}$-hard, i.e., unless $\mathcal{P} = \mathcal{NP}$, there is no polynomial-time algorithm that solves the problems such as traveling salesman and set covering [34]. Despite the negative theoretical result for the computational complexity, there is great interest in exploring special

structures of mathematical programs that enable the solution of realistic size instances of COP optimally in a reasonable time.

Suppose that we have a *mixed-integer program* (MIP) given by

$$\min \quad c^\top x \tag{1.2a}$$

$$\text{s.t.} \quad Ax \leq b \tag{1.2b}$$

$$x \in \mathbb{Z}_+^i \times \mathbb{R}_+^{n-i}, \tag{1.2c}$$

where x is a vector of n decision variables, $c \in \mathbb{R}^n$, $A \in \mathbb{R}^{m \times n}$, $b \in \mathbb{R}^n$, and m is the number of constraints. For the vector x, we assume that the first i components of x are restricted to be integral. The set $\{x \in \mathbb{Z}_+^i \times \mathbb{R}_+^{n-i} : Ax \leq b\}$ is called the *feasible region* of MIP. Note that if $i = n$, then formulation (1.2) is a *pure* integer program. A COP is typically formulated as a pure *binary* integer program, which has $i = n$ and the feasible region is $\{x \in \{0,1\}^n : Ax \leq b\}$ (or $\{x \in \mathbb{B}^n : Ax \leq b\}$). Here, we define $x_i = 1$ if the i^{th} component of x is selected, $x_i = 0$ otherwise.

In the real world, due to market fluctuations, different types of human behavior, weather conditions, and many other reasons, COPs often involve uncertainties, and some parameters are unknown before the decisions have to be made. We classify optimization problems according to the certainty of parameters. A *deterministic problem* has all parameters known with certainty, and a *stochastic problem* includes the uncertainty of some parameters. Taking the set covering problem as an example, given a collection of n subsets of a certain number of items, a deterministic set covering problem aims to select the minimum number of subsets from the collection, such that each item is covered by at least one selected subset. A stochastic (probabilistic) set covering problem assumes that when a subset is chosen, there is uncertainty in which items in the subset are actually covered. For such a case, the set of feasible subsets $\mathcal{F}$ is uncertain in problem (1.1). In this dissertation, we conduct mathematical optimization models and computational optimization techniques on stochastic COPs. Throughout this dissertation, we develop several solution methods based on the idea of *two-stage* stochastic programming. We provide a brief review of the classical two-stage stochastic programs in

the next subsection.

1.2 A Two-Stage Stochastic Program

We introduce a two-stage stochastic program, which models decision-making in consecutive stages, and involves two sets of decision variables. Let $(\Omega, E, \mathbb{P})$ be a probability space, where Ω is the sample space, E is the set of events, and $\mathbb{P}$ is the probability function. We assume that $(\Omega, E, \mathbb{P})$ is finite and the probability of an elementary event $\omega \in \Omega$ is $p_\omega := \mathbb{P}(\omega)$, where $\sum_{\omega\in\Omega} p_\omega = 1$. A general two-stage stochastic program is defined as

$$\min \quad c^\top x + \sum_{\omega\in\Omega} p_\omega f_\omega(x) \tag{1.3a}$$

$$\text{s.t.} \quad x \in \mathcal{X}, \tag{1.3b}$$

$$x \in X, \tag{1.3c}$$

where $c \in \mathbb{R}^{n_1}$, x is a vector of n_1 first-stage decision variables, the set $\mathcal{X}$ represents the constraints on the first-stage variables x, and the set X includes nonnegativity and continuous, integer or binary restrictions on x. Here, for all $\omega \in \Omega$, the function $f_\omega(x)$ is the objective function of the second-stage problem defined as

$$f_\omega(x) := \min \quad q(\omega)^\top y \tag{1.4a}$$

$$\text{s.t.} \quad y \in \mathcal{Y}(x, \omega) \cap Y, \tag{1.4b}$$

where $q(\omega) \in \mathbb{R}^{n_2}$, y is the vector of n_2 second-stage decision variables, the set Y imposes nonnegativity and integrality restrictions on y, and $\mathcal{Y}(x, \omega)$ represents the set of feasible second-stage decisions for a given first-stage solution x. The idea of the two-stage stochastic problem is that we make decisions in the first stage before the uncertainty is revealed. After the uncertainty is revealed in the second stage, we make operational decisions regarding the realization (scenario) of the uncertain parameters to minimize an objective function.

In stochastic COPs, we study a class of two-stage stochastic programs with $\mathcal{X} = \{\bar{A}x = b\}$, $X = \mathbb{B}^{n_1}$ and $\mathcal{Y}(x, \omega) = \{y \in Y : W(\omega)y = r(\omega) - T(\omega)x\}$, where $\bar{A} \in \mathbb{R}^{m_1\times n_1}$, $b \in \mathbb{R}^{m_1}$,

$W(\omega) \in \mathbb{R}^{m_2(\omega)\times n_2}$, $T(\omega) \in \mathbb{R}^{m_2(\omega)\times n_1}$, $r(\omega) \in \mathbb{R}^{m_2}$, m_1 denotes the number of constraints in the first-stage problem, and $m_2(\omega)$ denotes the number of constraints in the second-stage problem for $\omega \in \Omega$. The two-stage formulation (1.3) and (1.4) has a deterministic equivalent formulation

$$\min \quad c^\top x + \sum_{\omega\in\Omega} p_\omega q(\omega)^\top y_\omega \tag{1.5a}$$

$$\text{s.t.} \quad \bar{A}x = b, \tag{1.5b}$$

$$T(\omega)x + W(\omega)y_\omega = r(\omega), \qquad \omega \in \Omega, \tag{1.5c}$$

$$x \in X, \tag{1.5d}$$

$$y_\omega \in Y, \qquad \omega \in \Omega, \tag{1.5e}$$

where y_ω is the vector of second-stage decision variables under scenario ω for all $\omega \in \Omega$. A large-scale formulation (1.5) leads to computational challenges even for the state-of-the-art computer software. Furthermore, for stochastic COPs, the first-stage variables include binary decisions making formulation (1.5) even harder to solve.

In practice, we apply the Benders decomposition for solving formulation (1.5) with binary first-stage and continuous second-stage decisions. Next, we consider the first-stage master problem and the dual of the second-stage subproblem (1.4) with $Y = \mathbb{R}^{n_1}$ and $\mathcal{Y}(x,\omega) = \{y \in Y : W(\omega)y_\omega = r(\omega) - T(\omega)x\}$. The first-stage master problem is defined as

$$\min \quad c^\top x + \sum_{\omega\in\Omega} p_\omega \theta_\omega \tag{1.6a}$$

$$\text{s.t.} \quad \bar{A}x = b, \tag{1.6b}$$

$$(x,\theta) \in \mathcal{C}, \tag{1.6c}$$

$$(x,\theta) \in \mathcal{C}', \tag{1.6d}$$

$$x \in \mathbb{B}^n, \tag{1.6e}$$

where θ is a $|\Omega|$-dimensional vector of variables, θ_ω is the second-stage objective function approximation for scenario ω, and constraints (1.6c) and (1.6d) are the so-called *optimality*

and *feasibility cuts*, respectively. The dual of the second-stage subproblem is defined as

$$\max \quad \pi_\omega^\top (r(\omega) - T(\omega)x) \tag{1.7a}$$

$$\text{s.t.} \quad \pi_\omega^\top W(\omega) \leq q(\omega), \tag{1.7b}$$

where π_ω is the vector of second-stage dual variables.

In Algorithm 1, we describe a Benders decomposition (delayed constraint generation) algorithm for the two-stage programs. In the while loop of Algorithm 1, master problem (1.6) provides an incumbent solution $(\bar{x}, \bar{\theta})$ at each iteration (Line 4). We update all the second-stage problem (1.7) with the incumbent solution $\bar{x}$ and solve it for each $\omega \in \Omega$. For $\omega \in \Omega$, if the dual problem (1.7) is unbounded, and an extreme ray $\bar{d}(\omega)$ is identified, then $\bar{x}$ is infeasible, and a feasibility cut

$$\bar{d}(\omega)((r(\omega) - T(\omega)x)) \leq 0 \tag{1.8}$$

is added to the set $\mathcal{C}'$ of the master problem (1.6) (Line 9). For $\omega \in \Omega$, if the objective value of the dual problem (1.7) is finite and greater than $\bar{\theta}_\omega$, and an optimal solution $\bar{\pi}(\omega)$ is identified, then $\bar{x}$ is not optimal, and an optimality cut

$$\bar{\pi}(\omega)((r(\omega) - T(\omega)x)) \leq \theta_\omega \tag{1.9}$$

is added to the set $\mathcal{C}$ of the master problem (Line 13). Algorithm 1 terminates when no cuts are added to $\mathcal{C}'$ and $\mathcal{C}$. We refer to [17] for more details about the two-stage stochastic programs, and [12, 79] for solution algorithms. In addition, we refer to [43] for stochastic MIP programs.

In Chapter 2 of this dissertation, we consider a class of two-stage stochastic COP. The two-stage stochastic program described in this subsection is a risk-neutral stochastic programming model. This model enforces that all scenarios have to be satisfied by the first-stage incumbent solution $\bar{x}$, and considers the expected value of the second-stage costs. This requirement may lead to a conservative first-stage solution. In Chapter 3 of this dissertation, we also consider a risk-averse COP, where the risk is measured with a *chance constraint*. In risk-averse COP,

Algorithm 1: Delayed Constraint Generation Algorithm.

Start with $\mathcal{C}$ and $\mathcal{C}'$ (could be empty). ;
while $True$ **do**
  $CutAdded = 0$;
  Solve the master problem (1.6) and obtain an incumbent $(\bar{x}, \bar{\theta}_\omega)$. ;
  Update the dual objective function of problem (1.7) for all $\omega \in \Omega$. ;
  for $\omega \in \Omega$ **do**
    Solve subproblem (1.7) and obtain the optimal objective value f^*_ω of subproblem (1.7). ;
    if f^*_ω *is unbounded* **then**
      Add a feasibility cut (1.8) to $\mathcal{C}'$;
      $CutAdded = 1$
    end
    else if $f^*_\omega \geq \bar{\theta}_\omega$ **then**
      Add an optimality cut (1.9) to $\mathcal{C}$;
      $CutAdded = 1$
    end
  end
  if $CutAdded = 0$ **then**
    break;
  end
end
Output the solution $\bar{X} = \{i \in V : \hat{x}_i = 1\}$.

we can ignore some extreme scenarios and maintain the quality of solutions at a desired risk level. Next, we introduce this modeling paradigm.

1.3 A Chance-constrained Program

The main goal of a chance-constrained COP is to find the optimal subset of a finite set V to the problem, where the probability of an undesirable outcome is limited by a given risk level $\epsilon \in [0, 1]$. Let $\mathcal{B}(x)$ be a random event of interest for a given x. Given a risk level ϵ, the chance-constrained COP is defined as

$$\min\{c^\top x : \mathbb{P}(\mathcal{B}(x)) \geq 1 - \epsilon, x \in \mathcal{X} \cap \mathbb{B}^n\}, \tag{1.10}$$

where the set $\mathcal{X}$ is the deterministic constraints on the variables x, and $\mathbb{P}(\mathcal{B}(x)) \geq 1 - \epsilon$ represents the restriction that the probability of event $\mathcal{B}(x)$ is at least $1 - \epsilon$. Typically, the risk level ϵ is small such as $\epsilon = 0.05$ or 0.01.

In general, evaluating the probability of an event $\mathbb{P}(\mathcal{B}(x))$ for a given x is difficult due to multi-dimensional integrals. This challenge is generally overcome by sampling from a certain distribution [77, 83, 13, 67, 55]. Another challenge is that the feasible region of problem (1.10) is non-convex for general probability distributions. We deal with this challenge by formulating the feasibility condition using linear constraints. Luedtke and Ahmed [55] show that for general distributions with a large enough $|\Omega|\epsilon$, the sample-average approximation (SAA) can be applied to find solutions with good statistical lower bounds. For each scenario, problem (1.10) can be formulated as a deterministic MIP by introducing a binary variable and a big-M term for each inequality in the chance constraint [56]. An example of SAA is provided as follows. Suppose that an event $\mathcal{B}(x)$ can be linearized as $Ax \geq b(\omega)$, where $A \in \mathbb{R}^{m \times n}$, and $b(\omega)$ is an m-dimensional random vector parameterized by $\omega \in \Omega$. We consider a joint chance-constraint problem

$$\min\{c^\top x : \mathbb{P}(Ax \geq b(\omega)) \geq 1 - \epsilon, x \in \mathbb{B}^n\}. \tag{1.11}$$

In SAA, we generate a finite set of scenarios from the true distribution with $p_\omega := \mathbb{P}(\omega)$ for $\omega \in \Omega$. Given a set of scenarios Ω and $\epsilon \in [0, 1]$, a deterministic equivalent formulation of

problem (1.11) is

$$\min \quad c^\top x \tag{1.12a}$$

$$\text{s.t.} \quad Ax \geq b(\omega) - Mz(\omega), \qquad \omega \in \Omega \tag{1.12b}$$

$$\sum_{\omega \in \Omega} p_\omega z(\omega) \leq \epsilon \tag{1.12c}$$

$$x \in \mathbb{B}^n, \tag{1.12d}$$

$$z(\omega) \in \mathbb{B}, \qquad \omega \in \Omega, \tag{1.12e}$$

where $z(\omega)$ is a binary decision variable equal to 0 if in scenario ω event $\mathcal{B}(\omega)$ is observed, and 1 otherwise, and M is a large enough constant that makes constraint (1.12b) redundant when $z(\omega) = 1$. The constraint (1.12c) is the linear reformulation of the joint chance constraint $\mathbb{P}(Ax \geq b(\omega)) \geq 1 - \epsilon$.

SAA creates an approximation of the chance constraint, which can be evaluated for the given samples. In Chapter 3, we consider solution methods to obtain high-quality (ideally optimal) feasible solutions to chance-constrained COPs under a true (non-trivial) distribution assuming that there is an efficient oracle to check the feasibility of the chance constraint for a given solution. We establish exact methods that find a truly optimal solution of chance-constrained COPs. We show that the existence of the probability oracle allows us to derive stronger inequalities, which improve the computational time significantly. In addition, we show that the probability oracle may have a special structure that leads us to derive an efficient compact (polynomial-size) MIP formulation.

The chance-constrained COP ignores the worst $100\epsilon\%$ scenarios and ensures that the quality of solutions is maintained at a risk level. However, for a given solution, it does not consider the cost of the extreme scenarios. In some applications, the worst case information of a given solution is essential. For instance, in the stock market, it is important to understand how much money an investor will lose for the current portfolio in the worst case. In Chapter 4 of this dissertation, we consider a risk measure, the *conditional value-at-risk* (CVaR) at risk level ϵ, to evaluate the average of the losses that occur in $100\epsilon\%$ of the worst cases. In

the next subsection, we provide a brief review of CVaR and a class of risk-averse COPs that optimizes CVaR of a random objective function at a given risk level.

1.4 The Conditional Value-at-Risk

We measure a risk associated with a random variable through the conditional value-at-risk (CVaR) introduced by Artzner et al. [7], where larger values indicate less risky random outcomes, i.e., we prefer a higher profit of random outcomes in the stock market. In the above context, risk measures are referred to as *acceptability functionals*. More formally, let Z be a random variable under a certain distribution. Let η be a variable that approximates the *value-at-risk* (VaR) at a given risk level. The value-at-risk at a risk level $\epsilon \in (0, 1]$ is defined as

$$\mathrm{VaR}_\epsilon(Z) = \max\Big\{\eta : \mathbb{P}(Z \geq \eta) \geq 1 - \epsilon, \eta \in \mathbb{R}\Big\}. \tag{1.13}$$

The conditional value-at-risk at a risk level $\epsilon \in (0, 1]$ is defined as

$$\mathrm{CVaR}_\epsilon(Z) = \max\Big\{\eta - \frac{1}{\epsilon}\mathbb{E}([\eta - Z]_+) : \eta \in \mathbb{R}\Big\}, \tag{1.14}$$

where it gives the conditional expected value that is no larger than the value-at-risk at the risk level ϵ (Rockafellar and Uryasev [81]). In general, the risk level ϵ is small such as $\epsilon = 0.05$ or 0.01. Here, we assume that the probability space is finite, where $\Omega = \{\omega_1, \ldots, \omega_N\}$ and $\mathbb{P}(\omega_i) = p_i$ for $i = 1, \ldots N$. For all $i = 1 \ldots n$, we define z_i as the ith realization for the random variable Z. Let $[v]_+ = \max(v, 0)$ be the positive part of a number $v \in \mathbb{R}$. The optimization problem (1.14) with a finite number of scenarios can be formulated as

$$\begin{aligned}
\mathrm{CVaR}_\epsilon(Z) &= \max\Big\{\eta - \frac{1}{\epsilon}\mathbb{E}([\eta - Z]_+) : \eta \in \mathbb{R}\Big\} && \text{(1.15a)}\\
&= \max_{q \in [N]}\Big\{z_q - \frac{1}{\epsilon}\sum_{i \in [N]} p_i[z_q - z_i]_+\Big\} && \text{(1.15b)}\\
&= \max\Big\{\eta - \frac{1}{\epsilon}\sum_{i \in [N]} p_i w_i : w_i \geq \eta - z_i \quad \forall i \in [N], w \in \mathbb{R}^N_+, \eta \in \mathbb{R}\Big\}, && \text{(1.15c)}
\end{aligned}$$

where $[N] = \{1, \ldots, N\}$ represents the set of the first N positive integers and $z_q = \mathrm{VaR}_\epsilon(Z)$ for at least one $\omega_q \in \Omega$.

In Chapter 4, we consider a class of risk-averse COPs that evaluates the risk of a random objective function given by $f : 2^V \times \Omega \to \mathbb{R}$, where $f(S) : \Omega \to \mathbb{R}$ is defined by $f(S)(\omega) = f(S, \omega)$ for all $S \in F$ and $\omega \in \Omega$. We use the notation $f(x)$ for a given $x \in \mathbb{B}^{|V|}$ and $f(S)$ for the corresponding support $S \subseteq V$ interchangeably. Given a risk level $\epsilon \in (0, 1]$, a class of risk-averse COPs is defined as

$$\max\{\mathrm{CVaR}_\epsilon(f(x)) : x \in \mathcal{X}, x \in \mathbb{B}^n\}. \tag{1.16}$$

Under the assumption of the finite probability space, formulation (1.15c) is applied for solving problem (1.16). Given $\omega_i \in \Omega$, the mapping $f_i : 2^V \to \mathbb{R}$ is defined by $f_i(S) = f(S, \omega_i)$. Given $\epsilon \in (0, 1]$ and a finite set of scenarios Ω, we formulate problem (1.16) as

$$\max \quad \eta - \frac{1}{\epsilon} \sum_{i \in [N]} p_i w_i \tag{1.17a}$$

$$\text{s.t.} \quad w_i \geq \eta - f_i(x) \qquad \forall i \in [N] \tag{1.17b}$$

$$x \in \mathcal{X}, \tag{1.17c}$$

$$x \in \mathbb{B}^n, w \in \mathbb{R}^N_+, \eta \in \mathbb{R}. \tag{1.17d}$$

Note that if $f_i(x)$ is linear in x for all $\omega_i \in \Omega$ and $\mathcal{X} = \{Ax \leq b\}$, then formulation (1.17) is a MIP. Formulation (1.17) provides an approximation of problem (1.16). In Chapter 4, we consider solution methods to solve problem (1.16) under a certain distribution assuming that there is an efficient oracle to compute function $\mathrm{CVaR}_\epsilon(f(x))$ for a given x.

1.5 Research Scope and Outline

In the first part of this dissertation, we provide two-stage stochastic *submodular* optimization models, where the second-stage value function is submodular. A function $g : 2^V \to \mathbb{R}$ is submodular if $g(S \cup \{v\}) - g(S) \geq g(S' \cup \{v\}) - g(S')$ for $S \subseteq S' \subseteq V$ and $v \in V$. This problem is inspired by a stochastic *influence maximization* problem arising in social networks. Social networks connect different people and groups, providing a marketing platform that allows users to share their opinions via word-of-mouth. Marketers may have interests in

finding a small number of influential people to reach a large number of people in the social network. This problem can be formulated as the influence maximization problem that targets k individuals to influence the largest expected number of people in a social network. A social network is represented as a directed graph, where nodes represent individuals and an arc represents a potential influence relationship between individuals. We consider two stochastic diffusion models, the *independent cascade* and the *linear threshold* models, to represent the influence relationships between individuals. In real-world datasets, the size of social networks is usually large, and the connection of individuals is complex. There exists a greedy heuristic with a 63% optimality guarantee to obtain a solution to this problem. We show that the influence maximization problems are special cases of the two-stage stochastic submodular optimization model. We obtain an explicit form of the submodular inequalities and identify the facet-defining conditions of these cuts. We fill the 37% optimality gap by proposing a delayed constraint generation algorithm that solves the influence maximization problem optimally. Computational results demonstrate the effectiveness of our proposed method with large-scale real-world datasets.

In the second part of this dissertation, we study a class of chance-constrained combinatorial optimization problems inspired by a probabilistic partial set covering problem (PPSC). Recall that given a collection of n subsets of a certain number of items, a deterministic set covering problem aims to select the minimum cost of subsets from the collection, such that each item is covered by at least one selected subset. In PPSC, we assume that when a subset is selected, there is uncertainty in which items in the subset are covered. Given a fixed target number of items to cover, PPSC aims to find the minimum cost selection of subsets, which cover at least the target number of items with a high probability. In chance-constrained programs (CCPs), a challenge of evaluating the probability of an event is generally overcome by sampling from the true distribution. A given set of samples creates an approximation of the probability of the event; however, for such a sampling-based approach, the true feasibility of the approximate solution cannot be guaranteed. For example, consider an application to locate the minimum number of emergency vehicles that can reach an incident in a target

number of districts within r minutes with a high probability. In this problem, travel times are uncertain. An infeasible solution provided by the sampling-based approach indicates that the selected placement of emergency vehicles may not be able to guarantee a timely service to a potentially life-threatening emergency. To overcome this challenge, we consider situations under which there exists an efficient oracle provides an exact value of the probability of the desirable event for a given incumbent solution. We introduce the concept of a probability oracle for a class of combinatorial CCPs and propose a general method to solve such CCPs. We observe that under the independent cascade and linear threshold models, PPSC admits an efficient probability oracle. We show that based on this probability oracle, we can solve PPSC by using a compact deterministic MIP under the linear threshold. Furthermore, we propose a modified sampling-based method that utilizes the submodular property of the coverage function, where an efficient oracle can be a useful tool for checking and fixing the feasibility of a solution given by the sampling-based approach. We introduce a new class of facet-defining inequalities for the submodular substructure of PPSC. Our computational results demonstrate the effectiveness of our proposed method.

In the third part of this dissertation, we consider risk-averse solutions to stochastic submodular maximization problems inspired by a risk-averse set covering problem (RASC). Given an integer k, a set of items V_2 and a collection of n subsets $S_j \subseteq V_1, j \in V_1 := \{1, \ldots, n\}$ such that $\cup_{j=1}^{n} S_j = V_2$, a variant of set covering aims to choose k subsets from V_1 that covers the largest number of items in V_2. Suppose that there is uncertainty on whether a chosen subset can cover an item. Given a risk level ϵ, RASC aims to choose at most k subsets from the collection so that the expected value of the number of covered items with respect to the worst $100\epsilon\%$ of the outcomes is maximized. In this chapter, we consider a class of risk-averse submodular maximization problems (RASM), and consider RASC as a special case of RASM. Suppose that we have a random objective function, which is nondecreasing, submodular and higher values of the random objective function are preferred. We measure the risk associated with the random objective function through the conditional value-at-risk (CVaR). That is, given a risk level ϵ, we want to maximize the expected value of the worst

$100\epsilon\%$ of outcomes. In this context, risk measures are referred to as *acceptability functionals*. We use CVaR as the acceptability functional to model RASM that aims to maximize the CVaR of the random objective function at a given risk level. In this chapter, our goal is to solve RASM optimally. We assume that under certain distributions, there exists an efficient oracle, which provides an exact value of CVaR for a given incumbent solution. We give new valid inequalities and propose a delayed constraint generation algorithm that provides the optimal solution for RASM. Moreover, under the assumption of a finite probability space, we provide a two-stage stochastic programming model and another delayed constraint generation algorithm to solve RASM. Finally, taking RASC as an example of RASM, we observe that under the independent cascade and linear threshold models, RASC admits an efficient CVaR oracle. We demonstrate the proposed methods on RASC.

The remainder of this dissertation is organized as follows. In Chapter 2, we study two-stage stochastic programming method that exploits submodularity and its application to the influence maximization problem. In Chapter 3, we study a chance-constrained combinatorial optimization problem with a probability oracle and its application to probabilistic partial set covering. In Chapter 4, we study a class of risk-averse submodular maximization problems and its application to risk-averse set covering. Finally, we share our conclusions and future work in Chapter 5.

Chapter 2

A TWO-STAGE STOCHASTIC PROGRAMMING APPROACH FOR INFLUENCE MAXIMIZATION IN SOCIAL NETWORKS

2.1 Introduction

This chapter is based on [105]. The exploding popularity of social networking services, such as Facebook, LinkedIn, Google+ and Twitter, has led to an increasing interest in the effective use of word-of-mouth to market products or brands to consumers. A few individuals, seen as influencers, are targeted with free merchandise, exclusive deals or new information on a product or brand. Marketers hope that these key influencers promote the product to others in their social network through status updates, blog posts or online reviews and that this information propagates throughout the social network from peers to peers of peers until the product "goes viral." Therefore, a key question for marketers with limited budgets and resources is to identify a small number of individuals whom to target with promotions and relevant information so as to instigate a cascade of peer influence, taking into account the network effects.

2.1.1 Literature Review

Domingos and Richardson [27] first introduce the problem of finding which customers to target to maximize the spread of their influence in the social network. The authors propose a Markov random-field-model of the social network, where the probability that a customer is influenced takes into account whether her connections are influenced. After building this network, the authors propose several heuristics to identify which k individuals to target in a viral marketing campaign, where k is a user-defined positive integer. Kempe et al. [38] formalize the optimization problem and introduce two fundamental models to maximize the

influence spread in a social network: the *independent cascade* model and the *linear threshold* model. The authors show that the optimization problems are $\mathcal{NP}$-hard, assuming that there is an efficient oracle to compute the influence spread function. This seminal work spurred a flurry of research on social networks with over 4600 citations recorded by Google Scholar in August 2017. Wang et al. [101] show that calculating the influence spread function is $\#\mathcal{P}$-hard under the probabilistic assumptions of Kempe et al. [38]. Therefore, the independent cascade problem is $\#\mathcal{P}$-hard, and there are two sources of difficulty. First, the calculation of the influence spread function is hard because there is an exponential number of scenarios. This difficulty is overcome by using sampling. Second, the seed selection is combinatorial in nature, and requires the evaluation of an exponential number of choices. This difficulty is overcome by seeking heuristic solutions in the literature. We describe the results of the seminal paper by Kempe et al. [38] and the subsequent developments in Section 2.2.

The majority of the existing work on optimization-based methods for social network analysis focus on various aspects other than influence maximization (see the review by Xanthopoulos et al. [106]) with the exception of some recent work [78, 32], which develop mathematical programming approaches to solve *deterministic* weighted target set selection problems so that the total cost of influencing *all* nodes in a social network is minimized. Outside the influence maximization realm, the first class of problems studied is that of identifying the influential nodes of a network with respect to the nodes' centrality and connectivity. As an example of this class of problems, Arulselvan et al. [8] propose an integer programming formulation for the problem of identifying k nodes whose removal from a *deterministic* social network causes maximum fragmentation (disconnected components). The second class is that of clustering the nodes of a *deterministic* social network to identify the cohesive subgroups of the network. For example, Balasundaram et al. [10] and Ertem et al. [29] utilize optimization models to identify clique relaxations. Third, game-theoretic approaches are used to study various aspects of social networks, such as modeling competitive marketing strategies of two firms to maximize their market shares (see e.g., [16]).

In contrast to these models, we focus on the stochastic influence maximization prob-

lems and propose a two-stage stochastic programming method. In addition, by utilizing the submodularity of the second stage value (objective) function, we develop effective decomposition algorithms. Two-stage stochastic programming is a versatile modeling tool for decision-making under uncertainty. In the first stage, a set of decision needs to be made when some parameters are random. In the second stage, after the uncertain parameters are revealed, a second set of (recourse) decisions are made so that the expected total cost is minimized. We refer the reader to Birge and Louveaux [17] and Shapiro et al. [90] for an overview of stochastic (linear) programming. To the best of our knowledge, Song and Dinh [92] provide the only study besides ours that uses a stochastic programming approach to solve a problem in social networks. In this chapter, the authors consider the problem of protecting some arcs of a social network (subject to a limited budget) so that the damage caused by the spread of rumors from their sources to a set of targeted nodes is minimized.

2.1.2 Our contributions

Despite the ubiquity of social networks, there has been a paucity of research in finding provably optimal solutions to the two fundamental problems of maximizing influence in social networks (independent cascade and general threshold). The algorithms studied to date are approximation algorithms with a worst-case guarantee within 63% optimal ([39], and references therein). The proposed heuristics are tested on real social networks and compared to other simple heuristics. However, their practical performance has not been tested against the optimal solution due to the hardness of the problem and the unavailability of an algorithm that can find the optimal solution for large-scale instances of the problem. To fill this gap, we introduce a new class of problems that we refer to as two-stage stochastic submodular optimization models. We propose a delayed constraint generation algorithm to find the optimal solution to this class of problems with a finite number of samples. The proposed delayed constraint generation algorithm exploits the submodularity of the second-stage value function. The influence maximization problems of interest are special cases of this general problem class. Utilizing the special structure of the influence function, we give

an explicit characterization of the cut coefficients of the submodular inequalities, and identify conditions under which they are facet-defining for the full master problem that is solved by delayed constraint generation. This leads to a more efficient implementation of the proposed algorithm than is available from a textbook implementation of available algorithms for this class of problems [12, 98, 64]. In addition, we give the complete linear description of the master problem for $k = 1$, where k is the number of nodes targeted in a social network. We illustrate our proposed algorithm on the classical *independent cascade* and *linear threshold* problems [38]. In our computational study, we show that our algorithm outperforms a basic implementation of the greedy heuristic in most of the large-scale real-world instances.

We note that while we demonstrate our algorithms on the independent cascade and linear threshold models, our approach is more generally applicable to many other variants of the influence maximization problem studied previously in the literature. Furthermore, beyond social networks, there are other applications of identifying a few key nodes in complex networks for which our models are applicable. For example, Ostfeld and Salomons [72] consider the problem of locating costly sensors on the crucial junctures of the water distribution network to ensure water quality and safety by the early detection and prevention of outbreaks. The models could also be useful in the development of immunization strategies in epidemic models (see, e.g., [57]), and prevention of cascading failures in power systems (see, e.g., [33]). Furthermore, it also applies to more general stochastic optimization problems that have submodular second-stage value functions. For example, recently Contreras and Fernández [25] consider a *deterministic* hub location problem, and prove that the routing costs in the objective function are submodular. Using this observation, the authors employ the delayed constraint generation algorithm of Nemhauser and Wolsey [64] to solve the optimization problem more effectively than the existing models for this problem. Our proposed algorithm can be used to solve a *stochastic* extension of the hub location problem, where in the first stage, the hub locations are determined, and in the second stage, after the revelation of uncertain demand of multiple commodities, the optimal routing decisions are made. Hence, the general two-stage stochastic submodular optimization model and method that we introduce

in Section 2.3 has a potential broader impact beyond social networks.

2.1.3 Outline

In Section 2.2, we formally introduce the influence maximization problem and review the greedy algorithm of Kempe et al. [38]. In Section 2.3, we define a general two-stage stochastic submodular optimization model, and describe a delayed constraint generation algorithm that exploits the submodularity of the second-stage value function. We show that for $k = 1$, solving a linear program with a simple set of submodular optimality cuts and the cardinality restriction on the seed set guarantees an integer optimal solution. In Section 2.4, we consider the two fundamental influence maximization problems as defined by Kempe et al. [38], namely *independent cascade* and *linear threshold*. We show that for these special cases of the two-stage stochastic submodular optimization problems, we can obtain an explicit form of the submodular optimality cuts and identify conditions under which they are facet defining. In Section 2.5, we report our computational experience with large-scale real-world datasets, which show the efficacy of the proposed approach in finding optimal solutions as compared to the greedy algorithm.

2.2 Greedy Algorithm of Kempe et al. [38]

In this section, we describe the modeling assumptions of Kempe et al. [38], and overview the greedy hill-climbing algorithm proposed by these authors. Suppose that we are given a social network $G = (V, A)$, where $|V| = n, |A| = m$. The vertices represent the individuals, and an arc $(i, j) \in A$ represents a potential influence relationship between individuals i and j. Our goal is to select a subset of *seed* nodes, $X \subset V$, with $|X| \leq k < n$ to activate initially, so that the expected number of people influenced by X (denoted by $\sigma(X)$) is maximized, where k is a given integer. (Note that the original problem statement is to select exactly k nodes to activate. However, for the relaxation that seeks $|X| \leq k$ seed nodes that maximize influence, there exists a solution for which the inequality holds at equality.) The influence propagation is assumed to be *progressive*, in other words, once a node is activated it remains

active.

Kempe et al. [38] show that for various influence maximization problems, the influence function $\sigma(X)$ is nonnegative, monotone and submodular. Therefore, the influence maximization problem involves the maximization of a submodular function. The authors show that this problem is $\mathcal{NP}$-hard even if there is an efficient oracle to compute the influence spread function. However, using the results of Cornuéjols et al. [26] and Nemhauser et al. [65] that the greedy method gives a $(1 - \frac{1}{e})$-approximation algorithm for maximizing a nonnegative monotone submodular function, where e is the base of the natural logarithm, [38] establish that the greedy hill-climbing algorithm solves the influence maximization problem with a constant (0.63) guarantee, assuming that the function $\sigma(X)$ can be calculated efficiently. Recognizing the computational difficulty of calculating $\sigma(X)$ exactly, which involves taking the expectation of the influence function with respect to a finite (but exponential) number of scenarios, Kempe et al. [38] propose Monte-Carlo sampling, which provides a subset of equiprobable scenarios, Ω, of moderate size. Letting σ_ω denote the influence function for scenario $\omega \in \Omega$, we get $\sigma(X) = \frac{1}{|\Omega|} \sum_{\omega \in \Omega} \sigma_\omega(X)$. The basic greedy approximation algorithm of Kempe et al. [38] is given in Algorithm 2.

Subsequently, Wang et al. [101] formally show that calculating $\sigma(X)$ is $\#\mathcal{P}$-hard under the assumption of independent arc probabilities $\pi_{ij}, (i,j) \in A$. Therefore, Kempe et al. [39] propose a modification where an arbitrarily good approximation of $\sigma(X)$ is obtained in polynomial time by sampling from the true distribution. In particular, Kempe et al. [39] show that for a sample size of $\mathbb{O}\left(\frac{n^2}{\varepsilon^2} \ln(1/\alpha)\right)$, the average number of activated nodes over the sample is a $(1 \pm \varepsilon)$-approximation to $\sigma(X)$, with probability at least $1 - \alpha$.

Further algorithmic improvements to the greedy heuristic are given in the literature (see [39, 20] for an overview). Most notably, Borgs et al. [18] give a randomized algorithm for finding a $(1 - 1/e - \epsilon)$-approximate seed sets in $\mathcal{O}((m+n)\epsilon^{-3} \log n)$ time for any precision parameter $\epsilon > 0$. Note that this run time is independent of the number of seeds k. The authors show that the running time is close to the lower bound of $\mathbb{O}(m+n)$ on the time required to obtain a constant factor randomized approximation algorithm. The proposed

Algorithm 2: Greedy Approximation Algorithm of Kempe et al. [38].

```
Start with X = ∅ and a sample set of scenarios Ω;
while |X| ≤ k do
    For each node i ∈ V \ X, use the sample Ω to approximate σ(X ∪ {i});
    Add node i with the largest estimate for σ(X ∪ {i}) to X;
end
Output the set of seed nodes, X.
```

randomized algorithm has a success probability of 0.6, and failure is detectable. Therefore, the authors suggest repeated runs if failure is detected to improve the probability of success.

2.3 A General Two-Stage Stochastic Submodular Optimization Model and Method

In this section, we define a general two-stage stochastic submodular optimization model and outline a delayed constraint generation algorithm for its solution. Then, in Section 2.4, we describe how this general model and method is applicable to the influence maximization problems of interest.

Let $(\Omega, \mathcal{F}, \mathbb{P})$ be a finite probability space, where the probability of an elementary event $\omega \in \Omega$ is $p_\omega := \mathbb{P}(\omega)$. Consider a general two-stage stochastic binary program

$$\max \quad c^\top x + \sum_{\omega \in \Omega} p_\omega \sigma_\omega(x) \tag{2.1a}$$

$$\text{s.t.} \quad x \in \mathcal{X} \tag{2.1b}$$

$$x \in \mathbb{B}^n, \tag{2.1c}$$

where $c \in \mathbb{R}^n$ is a given objective vector, the set $\mathcal{X}$ represents the constraints on the first-stage variables x and $\sigma_\omega(x)$ is the objective function of the second-stage problem for scenario $\omega \in \Omega$ solved as a function of first-stage decisions given by

$$\sigma_\omega(x) := \max \quad q^\top y \tag{2.2a}$$

$$\text{s.t.} \quad y \in \mathcal{Y}(x, \omega). \tag{2.2b}$$

Here q is an objective vector of conformable dimension, y is the vector of second-stage decisions, and $\mathcal{Y}(x, \omega)$ defines the set of feasible second-stage decisions for a given first-stage vector x, and the realization of the uncertain outcomes given by the scenario $\omega \in \Omega$. We assume that $\sigma_\omega(x) : \{0,1\}^n \to \mathbb{R}$ is known to be a submodular function for each $\omega \in \Omega$, and refer to the optimization problem (2.1) as a *two-stage stochastic submodular optimization model*. It is well-known from the property of submodular functions that if $\sigma_\omega(x), \omega \in \Omega$ is submodular, then so is the second-stage value function $\sigma(x) = \sum_{\omega\in\Omega} p_\omega \sigma_\omega(x)$, which is a nonnegative (convex) combination of submodular functions. Furthermore, we assume that $\mathcal{Y}(x, \omega)$ is a non-empty set for each $x \in \mathcal{X}, \omega \in \Omega$, a property known as *relatively complete recourse* in stochastic programming.

Next we overview a delayed constraint generation approach to solve the two-stage program (2.1). The generic master problem at an iteration is formulated as

$$\max \quad c^\top x + \sum_{\omega\in\Omega} p_\omega \theta_\omega \tag{2.3a}$$

$$\text{s.t.} \quad x \in \mathcal{X} \tag{2.3b}$$

$$(x, \theta) \in \mathcal{C}, \tag{2.3c}$$

where θ is a $|\Omega|$-dimensional vector of variables θ_ω representing the second-stage objective function approximation for scenario ω, constraints (2.3c) represent the so-called *optimality cuts* generated until this iteration. The set of inequalities in $\mathcal{C}$ provides a piecewise linear approximation of the second stage value function, which is iteratively refined through the addition of the optimality cuts. (We will describe different forms of these inequalities in the following discussion.) Let $(\bar{x}, \bar{\theta})$ be the optimal solution to the master problem at the current iteration. Then for all $\omega \in \Omega$ we solve the subproblems (2.2) to obtain $\sigma_\omega(\bar{x})$. We add valid optimality cuts to $\mathcal{C}$ if $\bar{\theta}_\omega > \sigma_\omega(\bar{x})$ for any $\omega \in \Omega$, otherwise we deduce that the current solution $\bar{x}$ is optimal. The generic version of the delayed constraint generation algorithm is given in Algorithm 3. In this algorithm, ε is a user-defined optimality tolerance. The

particular implementation of Algorithm 3 depends on the method with which subproblems are solved to obtain $\sigma_\omega(\bar{x})$ (in line 5 of Algorithm 3), and the form of the optimality cuts added to the master problem (in line 10 of Algorithm 3). In this section, we explore the possibility of utilizing the submodularity of the second-stage value function in a two-stage stochastic programming problem. We discuss a natural alternative in Appendix A, which we use as a benchmark.

Algorithm 3: Delayed Constraint Generation Algorithm.

1 Start with $\mathcal{C} = \{0 \leq \theta_\omega \leq n, \omega \in \Omega\}$. Let LB=$-\infty$ and UB=∞;
2 **while** $UB - LB > \varepsilon$ **do**
3 Solve the master problem (2.3) and obtain $(\bar{x}, \bar{\theta})$. Let UB be the upper bound obtained from the optimal objective value of the master problem;
4 **for** $\omega \in \Omega$ **do**
5 Solve Subproblem (2.2) to obtain $\sigma_\omega(\bar{x})$;
6 **if** $\bar{\theta}_\omega > \sigma_\omega(\bar{x})$ **then**
7 Add an optimality cut to $\mathcal{C}$;
8 **end**
9 **end**
10 Let $\sigma(\bar{x}) = \sum_{\omega\in\Omega} p_\omega \sigma_\omega(\bar{x})$. **if** $LB < \sigma(\bar{x})$ **then**
11 Let LB $\leftarrow \sigma(\bar{x})$, and let $\hat{x} \leftarrow \bar{x}$ be the incumbent solution
12 **end**
13 **end**
14 Output the set of seed nodes $\bar{X} = \{i \in V : \hat{x}_i = 1\}$.

Nemhauser and Wolsey [64] give submodular inequalities to describe the maximum of a submodular set function (see also [66]). Consider the polyhedra $\mathcal{S}_\omega = \{(\theta_\omega, x) \in \mathbb{R} \times \{0,1\}^n : \theta_\omega \leq \sigma_\omega(S) + \sum_{j\in V\setminus S} \rho^\omega_j(S)x_j, \forall S \subseteq V\}$, and $\mathcal{S}'_\omega = \{(\theta_\omega, x) \in \mathbb{R} \times \{0,1\}^n : \theta_\omega \leq \sigma_\omega(S) - \sum_{j\in S} \rho^\omega_j(V \setminus \{j\})(1 - x_j) + \sum_{j\in V\setminus S} \rho^\omega_j(S)x_j, \forall S \subseteq V\}$ for $\omega \in \Omega$, where

$\rho_j^\omega(S) = \sigma_\omega(S \cup \{j\}) - \sigma_\omega(S)$ is the marginal contribution of adding $j \in V \setminus S$ to the set S.

Theorem 1. *(cf. [64]) For a submodular and nondecreasing set function $\sigma_\omega : 2^n \to \mathbb{R}$, $\bar{X}$, with a characteristic vector $\bar{x}$, is an optimal solution to $\max_{S \subseteq V : |S| \leq k}\{\sigma_\omega(S)\}$, if and only if $(\theta_\omega, \bar{x})$ is an optimal solution to $\{\max\ \theta_\omega : \sum_{j \in V} x_j \leq k, (\theta_\omega, x) \in \mathcal{S}_\omega\}$. Similarly for a submodular and nonmonotone set function $\sigma_\omega : 2^n \to \mathbb{R}$, $\bar{X}$, with a characteristic vector $\bar{x}$, is an optimal solution to $\max_{S \subseteq V : |S| \leq k}\{\sigma_\omega(S)\}$, if and only if $(\theta_\omega, \bar{x})$ is an optimal solution to $\{\max\ \theta_\omega : \sum_{j \in V} x_j \leq k, (\theta_\omega, x) \in \mathcal{S}'_\omega\}$.*

Therefore, we can adapt the algorithm of Nemhauser and Wolsey [66] given for deterministic submodular maximization problems to two-stage stochastic submodular optimization problems. Note that because there are exponentially many submodular inequalities, we cannot add all of them a priori to the formulation. Instead, we use a delayed constraint generation approach that adds the violated inequalities as needed and solves the resulting mixed-integer program by branch-and-cut (see e.g. [15] for an overview of the general form of a delayed constraint generation algorithm for linear programs). The proposed method takes the form of Algorithm 3. For a given first stage solution, $\bar{x}$, which is a characteristic vector of the set $\bar{X}$, and scenario $\omega \in \Omega$, we use the optimality cut

$$\theta_\omega \leq \sigma_\omega(\bar{X}) + \sum_{j \in V \setminus \bar{X}} \rho_j^\omega(\bar{X}) x_j, \tag{2.4}$$

if the second-stage value function $\sigma_\omega(x)$ is nondecreasing and submodular. If the second-stage value function $\sigma_\omega(x)$ is nonmonotone and submodular, then we use the optimality cut given by the inequality

$$\theta_\omega \leq \sigma_\omega(\bar{X}) - \sum_{j \in \bar{X}} \rho_j^\omega(V \setminus \{j\})(1 - x_j) + \sum_{j \in V \setminus \bar{X}} \rho_j^\omega(\bar{X}) x_j. \tag{2.5}$$

We refer the reader to Nemhauser and Wolsey [64] for validity of inequalities (2.4)-(2.5). Ahmed and Atamtürk [4] and Yu and Ahmed [107] strengthen the submodular inequalities by lifting, under the condition that the submodular utility function is strictly concave, increasing, and differentiable. These assumptions do not apply to our problem.

Corollary 1. *Algorithm 3 with optimality cuts* (2.4) *and* (2.5) *converges to an optimal solution in finitely many iterations for a two-stage stochastic program with binary first-stage decisions,* $x \in \{0,1\}^{|V|}$ *for which the second-stage value function,* $\sigma_\omega(x), \omega \in \Omega$, *(*$|\Omega|$ *finite) is submodular nondecreasing and submodular nonmonotone, respectively.*

Proof. The result follows from the fact that the number of feasible first stage solutions is finite, and from Theorem 1. □

Note that Algorithm 3 is generally applicable to two-stage stochastic programs with binary first-stage decisions, $x \in \{0,1\}^n$, where the second-stage value function, $\sigma_\omega(x)$ is submodular for all $\omega \in \Omega$. There is very limited reporting on the computational performance of this algorithm even for deterministic submodular maximization problems for which the method was originally developed (see [45, 25], for computational results on quadratic cost partition and hub location problems, respectively). Kawahara et al. [37] utilize the algorithm of Nemhauser and Wolsey [64] along with convexity cuts derived from Lovász extension of submodular functions to solve deterministic submodular maximization problems. The authors derive inequalities that are not globally valid as they cut off solutions that do not strictly improve upon the current incumbent solution. To the best of our knowledge, our work is the first adaptation and testing of this algorithm for two-stage stochastic optimization. While the submodular inequalities (2.4)-(2.5) are implicit in that they require the calculation of $\rho_j^\omega(\cdot)$ terms, in Section 2.4, we give an explicit form of the submodular optimality cuts for influence maximization problems of interest. This allows us to characterize conditions under which the optimality cuts are strong, and to improve the performance of a textbook implementation of the algorithm of Nemhauser and Wolsey [64].

Next, we consider the special case of cardinality-constrained first-stage problem (2.1), i.e., $\mathcal{X} := \{x \in \{0,1\}^n : \sum_{j \in V} x_j \leq k\}$, when $k = 1$. It is easy to see that in this case, the greedy algorithm is optimal. Note also that for fixed k, the problem is polynomially solvable (with respect to the input size of number of nodes, arcs and scenarios), because it involves evaluating $\mathcal{O}(n^k)$ possible functions $\sigma_\omega(X), \omega \in \Omega$. Observe that, without loss of generality,

we can assume that $p_\omega > 0$ for all $\omega \in \Omega$ (otherwise, we can ignore scenario ω), and that $\sigma_\omega(\emptyset) = 0$ (otherwise, we can add a constant to the influence function). Furthermore, because $\sigma_\omega(\cdot)$ is submodular $\rho_j^\omega(\emptyset) \geq \rho_j^\omega(S)$ for any $S \subseteq V, S \neq \emptyset$ and $j \in V \setminus S$. As a result, if $\rho_j^\omega(\emptyset) < 0$, then $x_j = 0$ in any optimal solution. Therefore, without loss of generality, we can assume that $\rho_j^\omega(\emptyset) \geq 0$ for all $j \in V, \omega \in \Omega$.

Proposition 1. *For submodular functions $\sigma_\omega(x), \omega \in \Omega$ with $\rho_j^\omega(\emptyset) > 0$ for all $j \in V, \omega \in \Omega$, and $\mathcal{X} := \{x \in \{0,1\}^n : \sum_{j \in V} x_j \leq 1\}$, adding the submodular optimality cut (2.4) with $\bar{X} = \emptyset$ to the linear programming (LP) relaxation of the master problem (2.3) for each $\omega \in \Omega$ is sufficient to give the (integer) optimal solution x^*.*

Proof. First, note that for $\bar{X} = \emptyset$, inequalities (2.4) and (2.5) are equivalent. Under the given assumptions, in an optimal solution $x \neq 0$, the right-hand side of (2.4) is positive for each $\omega \in \Omega$. Therefore, the decision variables $\theta_\omega > 0, \omega \in \Omega$, are basic variables at an extreme point optimal solution of the LP relaxation of the master problem (2.3). This gives us $|\Omega|$ basic variables, and the number of constraints is $|\Omega| + 1$. Hence, only one decision variable x_j for some $j \in V$ can be basic, and it is equal to 1 (due to constraint (2.6)), and $\theta_\omega = \rho_j^\omega(\emptyset) = \sigma_\omega(\{j\})$. Furthermore, this is the optimal solution to the master problem for the case $k = 1$. □

2.4 Application to the Stochastic Influence Maximization Problem

In this section, we specify how the general algorithm we propose for two-stage stochastic programs with submodular second-stage value functions applies to the influence maximization problems of interest. Kempe et al. [38] observe that even though the stochastic diffusion process of influence spread is dynamic, because the decisions of whom to activate do not influence the probability of an individual influencing another, we may envision the process to be static and ignore the time aspect. In other words, we can generate sample paths (scenarios) of likely events for each arc, a priori. As a result, the decision-making process considered by Kempe et al. [38] may be viewed as a two-stage stochastic program. In the first stage,

the nodes to be activated are determined. The uncertainty, represented by a finite collection of scenarios, Ω, is revealed with respect to how the influence spreads in the network. For each scenario $\omega \in \Omega$, with associated probability p_ω, let the influence spread given the initial seed set X be given by $\sigma_\omega(X) := |\{j \in V : \exists \text{ a path from } i \text{ to } j \text{ in } G_\omega, i \in X\}|$, i.e., $\sigma_\omega(X)$ is the number of vertices reachable from X in G_ω. As a result, the expected total influence spread of the initial seed set X is given by $\sigma(X) = \sum_{\omega\in\Omega} p_\omega \sigma_\omega(X)$. Let $x \in \{0,1\}^n$ be the characteristic vector of $X \subset V$. Where appropriate, we use $\sigma(x)$ interchangeably with $\sigma(X)$.

As observed by Kempe et al. [38], the influence function $\sigma_\omega(X)$ is submodular and monotone (nondecreasing) for various influence maximization problems. Then the two-stage stochastic programming formulation of the classical influence maximization problem is given by (2.1) where $c_j = 0$ for al $j \in V$ and the set $\mathcal{X}$ defines the cardinality constraint on the number of seed nodes given by

$$\sum_{j\in V} x_j \leq k, \tag{2.6}$$

for a given $0 < k < |V|$. Therefore, Algorithm 3 can be used to solve the influence maximization problem. Furthermore, note that the influence functions of interest in this chapter satisfy the assumption $\rho_j^\omega(\emptyset) > 0$ for all $j \in V, \omega \in \Omega$, because influencing only node j contributes at least one node (itself) to the influence function. In addition, the first-stage problem is cardinality-constrained. Hence Proposition 1 applies to the influence functions considered in this chapter.

To model the stochastic diffusion process and calculate the influence spread function, Kempe et al. [38] introduce a technique that generates a finite set, Ω, of sample paths (scenarios) by tossing biased coins. The coin tosses reveal, a priori, which influence arcs are active (live). A live-arc (i, j) indicates that if node i is influenced during the influence propagation process, then node j is influenced by it. For each scenario $\omega \in \Omega$, with a probability of occurrence p_ω, a so-called *live-arc graph* $G_\omega = (V, A_\omega)$ is constructed, where A_ω is the set of *live* arcs under scenario ω. Then the influence spread under scenario $\omega \in \Omega$ is calculated to obtain $\sigma_\omega(X)$. Hence, the expected influence spread function is given by

$\sigma(X) = \sum_{\omega \in \Omega} p_\omega \sigma_\omega(X)$. This is referred to as the "triggering model" or the "triggering set technique" by Kempe et al. [39]. The authors show the equivalence of the stochastic diffusion process of two fundamental influence maximization problems to the live-arc graph model with respect to the final active set. In addition, Kempe et al. [38] show that the influence spread in a live-arc graph representable problem is monotone and submodular under the given assumptions. As a result, our stochastic programming method applies to such problems. Next we describe the two fundamental influence maximization problems that are live-arc representable.

Independent Cascade Model: In the independent cascade model of Kempe et al. [38], it is assumed that each arc $(i, j) \in A$ of the social network $G = (V, A)$ has an associated probability of success, π_{ij}. In other words, with probability π_{ij} individual i will be successful at influencing individual j. We say that an arc (i, j) is *active* or *live* in this case. We generate a sample path (scenario) by tossing biased coins (with probability of π_{ij} for each arc $(i, j) \in A$) to determine whether the arc is active/live to construct the live-arc graph. Because each arc influence probability is independent, and does not depend on which nodes are influenced, Kempe et al. [38] show that the influence maximization problem is equivalent to maximizing the expected influence function in the live-arc graph model.

Linear Threshold Model: In the linear threshold model of Kempe et al. [38], each arc (i, j) in the social network $G = (V, A)$ has deterministic weight $0 \leq w_{ij} \leq 1$, such that for all nodes $j \in V$, $\sum_{i:(i,j)\in A} w_{ij} \leq 1$. In addition, each node $j \in V$ selects a threshold ν_j *uniformly at random.* A node is activated if sum of the weights of its *active* neighbors is above the thresholds, i.e., $\sum_{i:(i,j)\in A} w_{ij} x_i \geq \nu_j$. Given the set of initial seed nodes, $\bar{X}$, the activated nodes in the set U at time t influence their unactivated neighbor j at time $t + 1$ if $\sum_{u \in U} w_{uj} \geq \nu_j$. Kempe et al. [38] show that the linear threshold model also has an equivalent live-arc graph representation, where every node has at most one incoming live arc. Each node $j \in V$ selects at most one incoming live arc (i, j) with

probability w_{ij}, or it selects no arc with probability $1 - \sum_{i:(i,j)\in A} w_{ij}$. Given the seed set $\bar{X}$, Kempe et al. [38] prove the following two are equivalent:

1. The distribution of active nodes computed by executing the linear threshold model with starting seed set $\bar{X}$, and
2. the distribution of nodes reachable from $\bar{X}$ in the live-arc graph representation of the linear threshold model defined above.

Next, we demonstrate how the proposed algorithm (Algorithm 3) can be applied to influence maximization problems that have a live-arc graph representation. Subsequently, we give extensions where the proposed algorithm applies to models which are not live-arc graph representable. In such models, the form of the cuts change, but as long as the influence spread function is submodular, the proposed algorithm applies.

2.4.1 Exploiting the Submodularity of the Second-Stage Value Function for Live-Arc Graph Models

Utilizing Theorem 1, we give an explicit description of the submodular inequalities for the influence maximization problems that have live-arc graph representations. We say that a node j is reachable from a set of nodes S, in scenario $\omega \in \Omega$, if there exists a node $i \in S$ such that there is a directed path from i to j in the graph $G_\omega = (V, A_\omega)$. It is well known that reachability can be checked in linear time with respect to the number of arcs using depth- or breadth-first search. For $S \subseteq V$ and $\omega \in \Omega$, let $R(S)$ be the set of nodes reachable from the nodes in S not including the nodes in S, and let $\bar{R}(S)$ be the set of nodes not reachable from the nodes in S in the graph $G_\omega = (V, A_\omega)$.

Proposition 2. *For $S \subseteq V$ and $\omega \in \Omega$ the inequality*

$$\theta_\omega \leq \sigma_\omega(S) + \sum_{j\in\bar{R}(S)} r_j^\omega(S) x_j, \tag{2.7}$$

is a valid optimality cut for the master problem (2.3)*, where $r_j^\omega(S)$ is the number of nodes reachable from $j \in \bar{R}(S)$ (including j) that are not reachable from any node in S in G_ω.*

Proof. From Theorem 1, we know that $\theta_\omega \leq \sigma_\omega(S) + \sum_{j \in V \setminus S} \rho_j^\omega(S) x_j$ is a valid inequality. Note that $\bar{R}(S) \subseteq V \setminus S$ and for $j \in \bar{R}(S)$, we have $\rho_j^\omega(S) = r_j^\omega(S)$, in other words, the marginal contribution of adding $j \in \bar{R}(S)$ to S is precisely $r_j^\omega(S)$. Furthermore, for any node $j \in R(S)$, the marginal contribution of adding j to S is zero, because j is already reachable from at least one node in S. This completes the proof. □

We refer to the cuts in the form of (2.7) as *submodular optimality cuts*. Note that to obtain an inequality (2.7) for a given G_ω and S, we need to solve multiple reachability problems on the same graph, where time complexity of a single reachability problem by depth- or breadth-first search is $\mathcal{O}(|A_\omega|)$. We first compute $\sigma_\omega(S)$ by solving a reachability problem on G_ω and mark all nodes reachable from S. Then, for each $j \in \bar{R}(S) \setminus S$, we compute $r_j^\omega(S)$ by solving another reachability problem, where we count the number of unmarked nodes reachable from j. Hence the overall complexity of generating an inequality (2.7) is $\mathcal{O}(|A_\omega| \times |\bar{R}(S) \setminus S|)$.

Next we give conditions under which inequalities (2.7) are facet defining for $\text{conv}(\mathcal{S}_\omega)$.

2.4.2 Strength of the Submodular Inequalities

For $i \in V$, let $\text{indeg}(i)$ and $\text{outdeg}(i)$ denote the in-degree and out-degree of node i, respectively. Let $T := \{i \in V : \text{indeg}(i) = 0\}$, we refer to the nodes in T as *root nodes*. For $i \in V \setminus T$, let P_i be the set of root nodes such that i is reachable from the nodes in this set, i.e., $P_i := \{j \in T : i \in R(\{j\})\}$. Finally, let $L := \{i \in V : \text{indeg}(i) > 0, \text{outdeg}(i) = 0\}$ denote the set of *leaf nodes* that have no outgoing arcs.

First, note that the submodular inequality (2.7) for a set S is equivalent to that for the set $S \cup R(S) =: \hat{R}(S)$, because $\sigma_\omega(S) = \sigma_\omega(\hat{R}(S))$, $\bar{R}(S) = \bar{R}(\hat{R}(S))$, $r_j^\omega(S) = r_j^\omega(\hat{R}(S))$ for all $j \in \bar{R}(S)$ and $\rho_j^\omega(S) = 0$ for $j \in R(S)$. Therefore, in what follows, without loss of generality, we assume that for all non-leaf nodes $i \in S \setminus L$, we have $R(\{i\}) \subseteq S$ (Assumption **A1**).

Proposition 3. *For $S \subseteq V$ and $\omega \in \Omega$ the submodular inequality* (2.7) *is facet defining for conv($\mathcal{S}_\omega$) only if the following conditions hold*

1. *if $i \in S$, then $i \notin T$,*

2. *there exists $T' \subseteq T$ with $|T'| < k$ such that $S \subseteq R(T')$.*

These conditions are also sufficient

1. *if $S = \emptyset$ (for any $k \geq 1$), or*

2. *if $|S| = 1$ for $k \geq 2$.*

Proof. **Necessity**

1. Suppose, for contradiction, that there exists $i \in S \cap T$. Now consider the submodular inequality (2.7) for the set $S' = S \setminus \{i\}$ given by

$$\theta_\omega \leq \sigma_\omega(S') + \sum_{j \in \bar{R}(S')} r_j^\omega(S') x_j = \sigma_\omega(S) - 1 + x_i + \sum_{j \in \bar{R}(S)} r_j^\omega(S) x_j, \tag{2.8}$$

   which follows because the set of all descendants of i, $R(\{i\})$ is contained in S by Assumption **A1**, so removing i reduces the influence function by exactly 1 (recall that, by the contradictory assumption $i \in T$, hence its in-degree is 0 and it is not influenced by any other node in the graph), and the set of nodes not reachable from S' is given by $\bar{R}(S') = \bar{R}(S) \cup \{i\}$, and hence the coefficients $r_j^\omega(S') = r_j^\omega(S)$ for $j \in \bar{R}(S)$, and $r_i^\omega(S') = 1$. Because $x_i \leq 1$, inequality (2.8) dominates the submodular inequality (2.7) for this choice of S. Hence, the submodular inequality for a set S such that there exists $i \in S \cap T$ is not facet defining for conv($\mathcal{S}_\omega$).

2. Suppose, for contradiction, that there does not exist $T' \subseteq T$ with $|T'| < k$ such that $S \subseteq R(T')$. In other words, the minimum cardinality of root nodes $T' \subseteq T$ such that $S \subseteq R(T')$ is greater than or equal to k. In this case, consider the set

$\hat{S} := \{i \in S : \nexists j \in S \text{ with } i \in R(\{j\})\}$, in other words, $\hat{S}$ is the set of nodes in the graph induced by S that have no incoming arcs from other nodes in S. Note that from condition (i), we know that $\hat{S} \cap T = \emptyset$. Then, by the contradictory assumption, there exist at least k nodes, say nodes $1, \ldots, k \in \hat{S}$ such that $P_i \cap P_j = \emptyset$ for all pairs $i, j \in \{1, \ldots, k\}, i \neq j$. Now consider the submodular inequality (2.7) for the set $S' = S \setminus \{1, \ldots, k\}$ given by

$$\theta_\omega \leq \sigma_\omega(S') + \sum_{j \in \bar{R}(S')} r_j^\omega(S') x_j = \sigma_\omega(S) - k + \sum_{j \in \bar{R}(S)} r_j^\omega(S) x_j + \sum_{i=1}^{k} \sum_{j \in \hat{R}(P_i) \setminus R(\{i\})} x_j, \quad (2.9)$$

which follows because the set of all descendants of $i \in \{1, \ldots, k\}$, $R(\{i\})$, is contained in S by Assumption **A1**, so removing nodes $i = 1, \ldots, k$ reduces the influence function by exactly k, and the set of nodes not reachable from S' is given by $\bar{R}(S') = \bar{R}(S) \cup \{1, \ldots, k\}$. In addition, the coefficients $r_j^\omega(S') = r_j^\omega(S)$ for $j \in \bar{R}(S)$ such that $j \notin \cup_{i=1}^k \left(\hat{R}(P_i) \setminus R(\{i\})\right)$, $r_j^\omega(S') = r_j^\omega(S) + 1$ for $j \in \bar{R}(S)$ such that $j \in \cup_{i=1}^k \left(\hat{R}(P_i) \setminus R(\{i\})\right)$, and $r_i^\omega(S') = 1$ for $i = 1, \ldots, k$. Because $\sum_{i=1}^k \sum_{j \in \hat{R}(P_i) \setminus R(\{i\})} x_j \leq \sum_{j \in V} x_j \leq k$, inequality (2.9) dominates the submodular inequality (2.7) for this choice of S. Hence, there must exist $T' \subseteq T$ with $|T'| < k$ such that $S \subseteq R(T')$ for the submodular inequality (2.7) to be facet defining for $\text{conv}(\mathcal{S}_\omega)$.

Sufficiency First, note that for $\omega \in \Omega$, $\dim(\mathcal{S}_\omega) = n + 1$. Let $\mathbf{e}_i$ be a unit vector of dimension n whose ith component is 1, and other components are zero.

1. Note that when $S = \emptyset$, the necessity conditions are trivially satisfied. Consider the $n + 1$ affinely independent points: $(\theta_\omega, x)^0 = \mathbf{0}$, and $(\theta_\omega, x)^i = (\sigma_\omega(\{i\}), \mathbf{e}_i)$, for $i \in V$. These points are on the face defined by the inequality (2.7) for $S = \emptyset$. Hence inequality (2.7) for $S = \emptyset$ is facet-defining for $\text{conv}(\mathcal{S}_\omega)$.

2. Note that for $|S| = 1$, the necessity conditions imply that $S := \{j\}$ for some $j \in L$. Consider the $n + 1$ affinely independent points: $(\theta_\omega, x)^0 = (\sigma_\omega(\{j\}), \mathbf{0})$; $(\theta_\omega, x)^j = (\sigma_\omega(\{j\}), \mathbf{e}_j$ and $(\theta_\omega, x)^i = (\sigma_\omega(\{i, j\}), \mathbf{e}_j + \mathbf{e}_i)$, for $i \in V \setminus \{j\}$. The last set of points

is feasible because we have $k \geq 2$ in this case. These points are on the face defined by the inequality (2.7) for $S = \{j\}$. Hence inequality (2.7) for $S = \{j\}$ is facet-defining for conv($\mathcal{S}_\omega$).

□

Note that during the course of the algorithm, if a submodular inequality (2.7) corresponding to the seed set S does not satisfy the necessary conditions given in Proposition 9, then a stronger inequality can be constructed using the arguments in the proof of the proposition.

From Proposition 9 we see that inequalities (2.7) with $S = \emptyset$ are facets of conv($\mathcal{S}_\omega$) for any $k \geq 1$. We will also see their importance in our computational study. Similarly, inequalities (2.7) with $|S| = 1$ are facets of conv($\mathcal{S}_\omega$) for any $k \geq 2$. We note that more conditions are necessary for the inequalities (2.7) with $|S| = 2$ to be facets of conv($\mathcal{S}_\omega$). We illustrate this in the next example.

Example 1. *Consider the network in Figure 2.1 for a given scenario $\omega \in \Omega$ and let $k = 2$. From Proposition 9, inequalities* (2.7) *with $S = \emptyset$, and inequalities* (2.7) *with $S = \{j\}$, for $j = 4, \ldots, 9$ are facet-defining for conv($\mathcal{S}_\omega$). Inequalities* (2.7) *with $S = \{7, 8\}$ or $S = \{5, 6\}$ are facets of conv($\mathcal{S}_\omega$); each of these sets satisfies the necessary facet conditions in Proposition 9, which for these choices of S also turn out to be sufficient. However, the sets $S = \{7, 9\}$ or $S = \{4, 5\}$ satisfy the necessary facet conditions in Proposition 9, but they do not lead to facet-defining inequalities for $k = 2$. Finally, $S = \{4, 7\}$ violates the necessity condition (ii) of Proposition 9 (the minimum number of root nodes that can influence 4 and 7 is $2 = k$) and is not a facet.*

It is important to note that in the direct adaptation of the delayed constraint generation algorithm proposed by Nemhauser and Wolsey [66] to our problem, for a given solution $\bar{x}$ to the current master problem, one would use the submodular inequalities (2.7), where we let $S = \{i \in V : \bar{x}_i = 1\} =: \bar{X}$. From Proposition 1, we have that Algorithm 3 with optimality cuts (2.7) with $S = \bar{X}$ converges to an optimal solution in finitely many iterations for two-stage stochastic submodular maximization problems. However, note that at any iteration, the

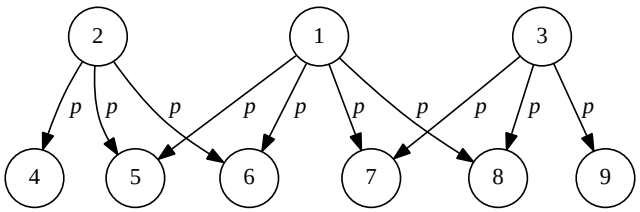

Figure 2.1: Network with 9 nodes and 10 arcs with equal influence probabilities p.

solution to the master problem, $\bar{x}$ will be such that $\sum_{j \in V} \bar{x}_j = k$. Therefore, all submodular optimality cuts (2.7) added will have $|S| = k$, and no facets with $S = \emptyset$ will be added for $k \geq 1$. This may lead to slow convergence of the delayed constraint generation algorithm with submodular optimality cuts for $S = \bar{X}$, which we illustrate next.

Example 1. *(Continued.) Consider the network in Figure 2.1, and suppose that* $|\Omega| = 1$*, hence we consider a deterministic problem. Adding the submodular optimality cut* (2.7) *for* $S = \emptyset$*:* $\theta_1 \leq 5x_1 + 4x_2 + 4x_3 + \sum_{j=4}^{9} x_j$ *to the cardinality constraint, and solving the linear programming relaxation of the master problem yields the integer optimal solution,* $\bar{x}_1 = 1, \theta_1 = 5$ *at the first iteration (from Proposition 1). In contrast, solving the master problem without any optimality cuts (i.e., with just the cardinality constraint) may lead to an initial solution of* $\bar{x}_2 = 1$*. Then the following set of submodular cuts are added in that order during the course of the algorithm of Nemhauser and Wolsey [66]:*

$$\theta_1 \leq 4 + 3x_1 + 4x_3 + \sum_{j=7}^{9} x_j \qquad (S = \bar{X} = \{2\}), \tag{2.10}$$

$$\theta_1 \leq 4 + 3x_1 + 4x_2 + \sum_{j=4}^{6} x_j \qquad (S = \bar{X} = \{3\}), \tag{2.11}$$

$$\theta_1 \leq 5 + 2x_2 + 2x_3 + x_4 + x_9 \qquad (S = \bar{X} = \{1\}). \tag{2.12}$$

Furthermore, none of the inequalities (2.10)-(2.12) *are facet-defining. Solving the LP relaxation of the master problem with the optimality cuts* (2.10)-(2.12) *leads to a fractional solution:* $\bar{x}_2 = \bar{x}_3 = 0.5$*. This small example highlights that a textbook implementation of the*

algorithm by Nemhauser and Wolsey [66] may lead to slow convergence because the algorithm (1) may explore, in the worst case, $O\binom{n}{k}$ many locally optimal solutions before finding an optimal solution, and (2) may require long solution times for each master problem, because the optimality cuts given by $S = \bar{X}$ may not be facet-defining and hence lead to weak LP relaxations.

These observations motivate us to devise a more efficient implementation of Algorithm 3, which we report in Section 2.5.

2.4.3 Extensions

In this section, we give various extensions of the influence maximization problems that can be solved using our proposed methods.

Extensions to live-arc graph models

Observe that while we demonstrate our general algorithm on the independent cascade and linear threshold models, our proposed model and method is applicable to many extensions of the social network problems studied in the literature. For example, an extension considered in the literature is to replace the cardinality constraint on the number of nodes selected with a knapsack constraint representing a marketing budget where each node has a different cost to market. This model also admits an adapted and more involved 0.63-factor greedy approximation algorithm (see, [40, 94]). In fact, our model is flexible enough to allow any constraints in $\mathcal{X}$ so long as the master problem can be solved with an optimization solver, while the greedy approximation algorithm needs careful adjustment and analysis for each additional constraint. Similarly, the time-constrained influence spread problem studied in Chen et al. [21] and Liu et al. [50] can also be solved using our method. In this problem, there is an additional constraint that the number of time periods it takes to influence a node should be no more than a given parameter τ. The resulting influence spread function is monotone and submodular, hence we can use inequalities (2.4) as the submodular optimality cuts.

Furthermore, we can efficiently calculate the coefficients $\rho_j^{\omega}(\bar{X})$ by solving, with breadth-first search, a modified reachability problem limiting the number of hops from the seed set $\bar{X}$ to any other node by τ.

General cascade and general threshold models

In the *general cascade model*, every node $j \in V$ has an *activation function* $p'_j(i, S) \in [0, 1]$ for $S \subseteq \{(k, j) \in A\} =: N^{in}(j)$ and $i \in N^{in}(j) \setminus S$. The activation function represents the probability that node j is influenced by node i given that the nodes in S failed to activate node i. The independent cascade model is a special case, where $p'_j(i, S) = \pi_{ij}$, independent of S.

In the *general threshold model*, every node $j \in V$ has an *threshold function* $f_j(S)$ for $S \subseteq N^{in}(j)$, where $f_j(\cdot)$ is monotone and $f_j(\emptyset) = 0$. As before, every node j selects a threshold ν_j uniformly at random in the range $[0, 1]$. Then, a node j is activated if for a given active set S, $f_j(S \cap N^{in}(j)) \geq \nu_j$. The linear threshold model is a special case, where $f_j(S) = \sum_{i \in S} w_{ij}$.

Kempe et al. [38] show that general cascade model is equivalent to the general threshold model with an appropriate selection of activation and threshold functions. This is not true for the independent cascade and the linear threshold models (see Example 2.14 in [20]). Furthermore, the influence spread function is no longer submodular. However, if $f_j(S)$ is submodular for all $j \in V$, then the influence spread is submodular (first conjectured by Kempe et al. [38] and later proven by Mossel and Roch [62, 63]). Therefore, the greedy hill climbing algorithm is a 0.63-approximation algorithm for this case as well. Algorithm 3 is applicable in the submodular threshold functions case, where the optimality cuts take the more general form (2.4) or (2.5) depending on the monotonicity of the function f.

2.5 Computational Experiments

In this section we summarize our experience with solving the influence maximization problem using the delayed constraint generation method (DCG) with various optimality cuts as given

in Algorithm 3, and the greedy hill-climbing algorithm (Greedy) of Kempe et al. [38] as given in Algorithm 2.

The algorithms are implemented in C++ with IBM ILOG CPLEX 12.6 Optimizer. All experiments were executed on a Windows Server 2012 R2 with an Intel Xeon E5-2630 2.40 GHz CPU, 32 GB DRAM and x64 based processor. In our implementation of Algorithm 3, we set the parameter $\varepsilon = 0$. For the master problem of the decomposition algorithm, the relative MIP gap tolerance of CPLEX was set to 1%, so a feasible solution which has an optimality gap of 1% is considered optimal.

2.5.1 Small-Scale Network

First, we study the quality of the solutions produced by DCG and Greedy on a small-scale network for which we can enumerate all possible outcomes of the random process. In these experiments, we are able to capture the random process precisely, and no information is lost through sampling from the true distribution. An illustrative network is given in Figure 2.1 with 9 nodes, 10 directed arcs and independent influence probability $\pi_{ij} = p$ for all $(i, j) \in A$. Our goal is to select $k = 2$ seed nodes, so that the objective value, which is the expected number of nodes influenced by the seed nodes, is maximized. We generate all possible influence scenarios (a total of $2^{10} = 1024$ scenarios). Note that under the assumption that each influence is independent of the others, the probability of scenario ω, which has $\ell \leq 10$ live arcs, is given by $p_\omega = (1-p)^{10-\ell} p^\ell$.

The solution of DCG and Greedy methods on 1024 scenarios with various values of $p = 0.1, 0.2, \ldots, 1$ is shown in Table 2.1. When $p \leq 0.5$, both algorithms have the same objective value. For $0.6 \leq p \leq 1$, Greedy selects node 1 as the seed in the first iteration of Algorithm 2 (line 4 of Algotihm 2) and selects either node 2 or 3 as the seed in the second iteration. However, DCG selects nodes 2 and 3 as the seed nodes, and provides a better objective value than Greedy (up to 12.5% improvement). So while Greedy does better than its worst-case bound (63%), it is within 12.5% of optimality.

Next, instead of generating all 1024 scenarios, we employed Monte-Carlo sampling, and

independently sampled different number of scenarios $|\Omega| = 10, 50$ and 100 according to different p values, and let $p_\omega = 1/|\Omega|$. We summarize the results of this experiment in Table 2.2. For eight out of 15 cases, DCG has a higher objective value than Greedy, and in all other cases Greedy attains the optimal objective value (mostly for small influence probabilities $p = 0.1, 0.3$). We also observe that the objective value for the instances with a larger number of scenarios is generally closer to the objective value with all 1024 scenarios (except for $p = 0.1$ and $|\Omega| = 100$). Note that Greedy is a 0.63-approximation algorithm even for the sampled problem, which assumes that the true distribution is given by the scenarios in Ω, whereas DCG provides the optimal solution to the sampled problem.

Table 2.1: Expected influence obtained from two algorithms for the small-scale network with 1024 scenarios.

	Objective values with different p									
Algorithm	$p = 1.0$	$p = 0.9$	$p = 0.8$	$p = 0.7$	$p = 0.6$	$p = 0.5$	$p = 0.4$	$p = 0.3$	$p = 0.2$	$p = 0.1$
DCG	8	7.4	6.8	6.2	5.6	5	4.48	3.92	3.32	2.68
Greedy	7	6.68	6.32	5.92	5.48	5	4.48	3.92	3.32	2.68

2.5.2 Large-Scale Network with Real-World Datasets

To evaluate the efficiency of DCG and Greedy on large networks, we conduct computational experiments on four real-world datasets with different categories and scales. These datasets are summarized below and in Table 2.3:

UCI-message is the dataset for the online student community network at the University of California, Irvine [71]. The 1,899 nodes represent the students, and the 59,835 directed arcs between two nodes indicate that one student sent a message to the other.

P2P02 is the dataset for the Gnutella peer-to-peer file sharing network from August 2002 [47, 80]. The 10,876 nodes represent the hosts, and the 39,994 undirected edges denote

Table 2.2: Expected influence obtained from two algorithms for the small-scale network with $|\Omega|$ scenarios.

		Objective values for different p				
$\|\Omega\|$	Algorithm	$p = 0.9$	$p = 0.6$	$p = 0.5$	$p = 0.3$	$p = 0.1$
10	DCG	7.1	5.2	4.7	3.4	2.6
10	Greedy	6.8	5.1	4.6	3.4	2.6
50	DCG	7.4	5.84	5.18	3.98	2.68
50	Greedy	6.82	5.66	5.06	3.98	2.68
100	DCG	7.38	5.61	5.04	3.96	2.76
100	Greedy	6.69	5.52	5.04	3.96	2.76

the connections between two hosts.

Phy-HEP is the dataset for the academic collaboration network in the "high energy physics theory" (HEPT) section of the e-print arXiv (www.arxiv.org) [38, 22, 39]. The 15,233 nodes represent the authors, and the 58,891 undirected edges represent the co-authorship between each pair of authors in the "high energy physics theory" papers from 1991 to 2003. Note that this is the original dataset considered in Kempe et al. [38], and it is commonly used as a benchmark in comparing various algorithms for maximizing influence in social networks.

Email-Enron is the dataset for the email communication network of the Enron Corporation [49, 41]. It is posted to the public by the Federal Energy Regulatory Commission during the investigation. The 36,692 nodes represent the different email addresses, and the 183,831 directed arcs denote one address sent a mail to the other.

Note that if the graph in the original datasets contain undirected edges between i and j, then we construct a directed graph with two directed arcs from i to j and j to i. We follow

the data generation scheme of Kempe et al. [38] in constructing live-arc graphs for these instances (i.e., the probabilities of influence on each arc for the independent cascade model, and the arc weights and node thresholds for the linear threshold model follow from Kempe et al. [38]).

In our experiments in this subsection, we compare three algorithms: *Greedy* is the greedy hill-climbing algorithm (Algorithm 2); *DCG-SubIneqs* is Algorithm 3 using submodular optimality cuts (2.7) for $S = \bar{X}$, where $\bar{X}$ is the optimal solution given by the master problem in the current iteration; and *DCG-SubWarmup* adds the submodular inequalities (2.7) for $S = \emptyset$ for each scenario, before the execution of DCG-SubIneqs as a warm-start. Proposition 1 shows that the submodular optimality cut (2.7) with $S = \emptyset$, which is referred to as EmptySetCut, is sufficient to find the optimal solution for $k = 1$ (note that $\rho_j^\omega(\emptyset) \geq 1$ for all $j \in V, \omega \in \Omega$, hence the assumptions of the proposition are satisfied). Since $k = 1$ is an easy case for DCG-SubWarmup due to Proposition 1, we include DCG-SubWarmup to test if EmptySetCut is also useful for $k > 1$. To verify this, we add EmptySetCuts for all scenarios to the master problem before executing the DCG algorithm, and solving the initial master problem. Note that the total computation time in our experiments includes the generation time of all EmptySetCuts, and the total number of user cuts also includes the number of EmptySetCuts. We also implemented Algorithm 3 using alternative optimality cuts (adapted and strengthened versions of integer-L-shaped cuts of [44], referred to as Benders-LC, and described in Appendix A); however, the running time of Benders-LC is extremely long. Therefore, we only report our results with DCG-SubIneqs, DCG-SubWarmup and Greedy, and discuss the inefficiency of Benders-LC in Appendix. B.

Independent Cascade Model

For the independent cascade model, we assign uniform influence probability $\pi_{ij} = p = 0.1$ independently to each arc (i, j) in the network as was done in Kempe et al. [38]. Note that Kempe et al. [38] consider the dataset **Phy-HEP** with influence probabilities $\pi_{ij} \in \{0.01, 0.1\}$ for each arc (i, j) in the network. However, we observe that for $\pi_{ij} = 0.01$,

Table 2.3: The summary of real world datasets.

	Dataset			
	UCI-message	P2P02	Phy-HEP	Email-Enron
Network Category	Online-Message	File-Shearing	Collaboration	Communication
Nodes	1,899	10,876	15,233	36,692
Edges	59,835	39,994	58,891	183,831
Format	Directed	Undirected	Undirected	Directed

the total number of live arcs is very small, resulting in sparse live-arc graphs with a large number of singletons. For example, the expected number of live arcs in our largest dataset **Email-Enron** is $183,831 \times 0.01 = 1,838$ with $p = 0.01$. However, the number of nodes of **Email-Enron** is 36,692, resulting in over 30,000 singletons in the network. Therefore, we focus on the more interesting case of $\pi_{ij} = 0.1$ in our experiments in this section.

We generate $|\Omega| = 100, 200, 300$ and 500 scenarios to find $k = 2$ to 5 seed nodes that maximize influence. Tables 2.4-2.5 summarize our experiments with the algorithms DCG-SubIneqs, DCG-SubWarmup and Greedy for the independent cascade model. Column "k" denotes the number of seed nodes to be selected. Column "Cuts(#)" reports the total number of submodular inequalities (2.7) added to the master problem of DCG-SubIneqs, and column "Time(s)" reports the solution time in seconds. We do not report the objective values in these experiments, because we are able to prove that despite its worst-case performance guarantee of 63%, Greedy is within the optimality tolerance for these instances. In Kempe et al. [38] Greedy is tested empirically against other heuristics such as choosing the nodes with k highest degrees in the graph G, because it is said that an optimal solution is not available. Therefore, our computational experiments also provide an empirical test on the performance of greedy heuristic when an optimal solution is available (to the sampled problem) due to our proposed method.

Table 2.4: Independent Cascade Model for UCI-message and P2P02

Datasets	k	$\lvert\Omega\rvert$	DCG-SubIneqs Time(s)	DCG-SubIneqs Cuts(#)	DCG-SubWarmup Time(s)	DCG-SubWarmup Cuts(#)	Greedy Time(s)
UCI-message	2	100	75	200	72	200	40
	3	100	82	200	78	201	50
	4	100	76	200	71	200	59
	5	100	81	200	73	200	69
	2	200	90	399	85	400	77
	3	200	91	400	88	400	96
	4	200	95	400	84	400	118
	5	200	89	400	87	400	138
	2	300	356	596	260	600	133
	3	300	274	600	264	600	168
	4	300	268	600	282	600	201
	5	300	273	600	272	600	232
	2	500	237	994	201	1000	202
	3	500	221	999	207	1000	252
	4	500	219	1000	203	1000	299
	5	500	211	1000	200	1000	347
	Average		171.1	549.3	157.9	550.1	155.1
P2P02	2	100	466	200	505	212	364
	3	100	463	216	514	216	520
	4	100	721	245	591	244	682
	5	100	572	232	547	232	823
	2	200	526	400	514	400	558
	3	200	553	412	552	411	794
	4	200	569	433	552	428	1014
	5	200	785	451	590	451	1227
	2	300	240	600	246	600	1852
	3	300	246	600	251	600	2652
	4	300	324	658	295	643	3452
	5	300	310	678	302	658	4369
	2	500	522	1000	515	1000	4796
	3	500	489	1000	522	1000	6922
	4	500	609	1075	721	1193	9018
	5	500	608	1131	620	1129	11043
	Average		500.2	583.2	489.8	588.6	3130.4

Table 2.5: Independent Cascade Model for Phy-HEP and Email-Enron

Datasets	k	$\|\Omega\|$	DCG-SubIneqs Time(s)	DCG-SubIneqs Cuts(#)	DCG-SubWarmup Time(s)	DCG-SubWarmup Cuts(#)	Greedy Time(s)
Phy-HEP	2	100	650	208	623	200	1141
	3	100	777	301	771	300	1654
	4	100	1064	385	1054	385	2107
	5	100	375	491	366	491	2530
	2	200	261	400	257	401	1593
	3	200	524	599	455	598	2185
	4	200	2208	770	1983	771	2770
	5	200	845	989	733	988	3344
	2	300	414	609	504	600	1123
	3	300	648	902	567	900	1561
	4	300	829	1040	863	1038	2004
	5	300	911	1190	737	1190	2465
	2	500	603	1000	614	1005	6250
	3	500	1266	1498	1353	1500	8992
	4	500	1544	1985	1434	1985	11545
	5	500	2128	2220	2315	2323	13808
	Average		940.4	911.7	914.9	917.2	4067
Email-Enron	2	100	1656	200	2004	200	7332
	3	100	1618	200	1721	200	9860
	4	100	1715	200	1618	200	12465
	5	100	1622	200	1628	200	15137
	2	200	4279	400	4262	400	11623
	3	200	3372	400	3420	400	15576
	4	200	3273	400	3668	400	19414
	5	200	3265	400	3529	400	23124
	2	300	5256	600	5037	600	17675
	3	300	4890	600	5003	600	24357
	4	300	4921	600	6116	726	31387
	5	300	4900	600	5000	600	38385
	2	500	9176	1000	8756	1000	25970
	3	500	9726	1000	9305	1000	34775
	4	500	9187	1000	9507	1000	43588
	5	500	11056	1000	9390	1000	52548
	Average		4994.5	550	4997.8	557.9	23951

From column Cuts(#) in Tables 2.4-2.5, we observe that the number of cuts added to the master problem generally increases with the number of seed nodes k. In other words, more iterations are needed to prove optimality if we have more seed nodes to select. Columns DCG-SubIneqs Time(s) and Cuts(#) show that the overall running time does not necessarily increase with the number of user cuts, as more cuts may help the master problem converge to an optimal solution faster. Recall that the running time of DCG-SubIneqs includes the solution time of the master problem (a mixed-integer program) and the cut generation time of submodular inequalities, which decomposes by each scenario.

From column Greedy Time(s) we see that, for the same size of scenarios, the running time of Greedy increases linearly as the number of seed nodes increases. Recall that DCG solves a mixed integer master problem after the cut generation phase, which makes its running time nonlinear as we observe from the columns DCG-SubIneqs Time(s) and DCG-SubWarmup Time(s). The majority of the overall time for DCG is spent on cut generation for most instances (for example, the cut generation takes, on average, 80% of the time over all four problem instances with $|\Omega| = 300$). Because we observe that the major bottleneck is the cut generation time and most of the remaining time is spent on the solution of the mixed-integer master problem, we did not implement other enhancements known to improve convergence of related Benders methods, such as the trust region method or heuristics (cf. [85, 70]).

Considering the average solution time, we observe that there is not much difference between DCG-SubWarmup and DCG-SubIneq, and DCG outperforms Greedy for large networks with more than 10,000 nodes. For example, for the instance Email-Enron in Table 5, the average solution time of Greedy is five times that of DCG-SubIneqs. Only for the smallest instance, UCI-message, Greedy is the fastest algorithm (see Table 4).

Linear Threshold Model

In this section, we summarize our experiments with the linear threshold model. Recall that in the live-arc graph representation of linear threshold models, at most one incoming arc is chosen for each node in the live-arc graph construction for each scenario. As in Kempe et al.

[38] we let the deterministic weight on each arc $(i, j) \in A$ be $w_{ij} = 1/\text{indeg}(j)$. We generate $|\Omega| \in \{100, 200, 300, 500\}$ for the four real-world datasets described earlier.

The results are shown in Tables 2.6-2.7. Similar to the independent cascade model, the running time of Greedy increases linearly in k. As in the previous experiments for the independent cascade model, DCG-SubWarmup is slower than Greedy only for the smallest dataset with fewer than 2,000 nodes (UCI-message) (see Table 2.6). For the large-scale datasets with over 10,000 nodes (P2P02, Phy-HEP, and Email-Enron) reported in Tables 2.6-2.7, we observe that the warm-up strategy is highly effective. It provides the best solution times, and fewer iterations and cuts. For example, DCG-SubWarmup outperforms Greedy by a factor of 2.46 in P2P02 in Table 2.6, a factor of 4.12 in Phy-HEP in Table 2.7, and a factor of 27 in Email-Enron in Table 2.7, the largest dataset considered.

Some comments are in order for both the independent cascade and linear threshold models. First, we make some observations on increasing k. As can be seen from our experiments, the running time of Greedy increases linearly with k. However, the increase in the running time of DCG is nonlinear, as can be expected. Hence, as we increase k, we need to set some time limits for both DCG and Greedy (currently, we impose no time limits). In this case, with DCG, we are still able to obtain an incumbent solution with k seed nodes and an estimate on optimality gap provided by the bound from the DCG master. However, with a time limit, we will have to stop Greedy prematurely, before it identifies all k seed nodes. For example, for the independent cascade model for the medium-sized instance P2P02, setting a time limit of one hour, for $k = 30$ and $|\Omega| = 300$, Greedy stops at time limit with a solution that has $k = 4$ seed nodes (see Table 2.4). On the other hand, DCG stops with an incumbent solution that has $k = 30$ seed nodes and an optimality gap of 3.7%. For the largest instance Enron, from Table 2.5, we observe that Greedy cannot even find the first seed node in over three hours. Similarly, as we increase $|\Omega|$, the running time of both algorithms increase greatly, and as in the case of increasing k we will have to impose time limits and stop Greedy prematurely to be able to compare the performance of the algorithms.

Note that, throughout the chapter, we present a so-called multicut version of the DCG

Table 2.6: Linear Threshold Model for UCI-message and P2P02

Datasets	k	$\lvert\Omega\rvert$	DCG-SubIneqs Time(s)	DCG-SubIneqs Cuts(#)	DCG-SubWarmup Time(s)	DCG-SubWarmup Cuts(#)	Greedy Time(s)
UCI-message	2	100	72	299	43	120	1
	3	100	94	378	70	169	2
	4	100	130	455	93	257	3
	5	100	162	606	89	303	4
	2	200	220	722	91	259	3
	3	200	221	735	80	296	4
	4	200	243	892	161	502	6
	5	200	469	1435	289	893	7
	2	300	188	590	200	513	5
	3	300	192	594	159	449	8
	4	300	573	1507	249	637	11
	5	300	543	1554	387	1011	13
	2	500	462	988	84	559	6
	3	500	747	1403	181	740	9
	4	500	665	1415	130	723	12
	5	500	1282	2909	462	1933	15
	Average		391.6	1030.1	173	585.3	6.8
P2P02	2	100	48	200	26	103	222
	3	100	55	200	28	108	337
	4	100	136	338	28	120	449
	5	100	230	580	31	133	565
	2	200	387	400	202	202	211
	3	200	583	596	225	214	311
	4	200	590	596	204	213	414
	5	200	1260	1156	306	290	512
	2	300	1564	600	584	308	1231
	3	300	1853	885	684	343	1833
	4	300	3529	1687	855	410	2531
	5	300	9557	4000	1656	765	3118
	2	500	1721	1000	576	503	1028
	3	500	1224	1000	534	512	1507
	4	500	4799	3308	600	552	1969
	5	500	10505	5861	1047	887	2415
	Average		2377.6	1403.6	474.1	353.9	1165.8

Table 2.7: Linear Threshold Model for Phy-HEP and Email-Enron

Datasets	k	$\|\Omega\|$	DCG-SubIneqs Time(s)	DCG-SubIneqs Cuts(#)	DCG-SubWarmup Time(s)	DCG-SubWarmup Cuts(#)	Greedy Time(s)
Phy-HEP	2	100	528	243	175	101	1116
	3	100	1117	428	333	182	1682
	4	100	1351	474	383	185	2235
	5	100	786	387	63	209	2740
	2	200	1141	586	477	359	1176
	3	200	1841	964	540	362	1791
	4	200	1448	779	602	363	2444
	5	200	1190	581	722	377	3229
	2	300	399	600	445	303	692
	3	300	489	600	516	312	1017
	4	300	943	871	1089	548	1329
	5	300	6425	3413	824	619	1642
	2	500	1910	1218	1229	887	4002
	3	500	2440	1421	1312	891	6342
	4	500	4340	2283	1635	988	8637
	5	500	9291	3845	2051	1040	10993
	Average		2227.4	1168.3	774.8	482.9	3191.7
Email-Enron	2	100	206	200	97	100	3972
	3	100	196	200	131	106	5814
	4	100	414	379	158	137	7605
	5	100	644	535	299	230	9366
	2	200	1240	400	608	200	4732
	3	200	419	400	223	215	7001
	4	200	642	593	285	265	9192
	5	200	902	760	447	378	11330
	2	300	740	600	335	300	9283
	3	300	666	600	375	326	13602
	4	300	1082	886	436	365	17833
	5	300	1975	1441	982	710	22354
	2	500	1199	1000	613	500	12113
	3	500	1192	1000	675	523	17858
	4	500	2446	1929	845	669	23443
	5	500	3490	2373	1044	844	28876
	Average		1090.8	831	472.1	366.8	12773.4

algorithm and its variants, where we add an optimality cut for each scenario at each iteration. We have also tested a single cut implementation, in which multiple cuts across all scenarios are aggregated into a single cut at each iteration. We observe a degraded performance of the single cut version for our problem instances; therefore, we present our results for the multicut approach. In particular, for the independent cascade model, the total computational time of the single cut version of DCG-SubWarmup is between 20 to 100% higher than the multicut version in all four datasets with 300 scenarios for $k > 1$. (See page 167 of [17], for a discussion on the problem-dependent nature of the performance of the single vs. multicut approach).

We remark that we implemented the basic greedy algorithm as proposed by Kempe et al. [38]. This implementation needs to perform the influence spread function evaluation for each node for each scenario at each iteration, resulting in a heavy running time of $\mathcal{O}(kn|\Omega|m)$. Several improvements to this implementation using different data structures and algorithmic enhancements have been proposed (see, e.g., [48, 22]). In particular, Chen et al. [22] propose the so-called Discount Greedy method and report that it solves the Phy-HEP instances within a second. In this chapter, our main goal is not to compete with these efficient heuristics. Instead, we focus on solving the problem on large-scale datasets optimally in a reasonable time. Our computational experiments demonstrate an overlooked opportunity to use optimization methods to solve stochastic influence maximization problems to provable optimality.

Chapter 3

CHANCE-CONSTRAINED COMBINATORIAL OPTIMIZATION WITH A PROBABILITY ORACLE AND ITS APPLICATION TO PROBABILISTIC PARTIAL SET COVERING

3.1 Introduction

This chapter is based on [104]. Chance-constrained programs (CCPs), first introduced in [19], aim to find the optimal solution to a problem such that the probability of satisfying certain constraints is at least at a certain confidence level. In this chapter, we consider chance-constrained *combinatorial* optimization problems. Given a vector of n binary decision variables $x \in \mathbb{B}^n$, we define $x_i = 1$ if the i^{th} component of x is selected, $x_i = 0$ otherwise. Let $\mathcal{B}(x)$ represent a random event of interest for a given x. Given a risk level $\epsilon \in [0, 1]$, a chance-constrained program is

$$\min\{b^\top x : \mathbb{P}(\mathcal{B}(x)) \geq 1 - \epsilon, x \in \mathcal{X} \cap \mathbb{B}^n\}, \tag{3.1}$$

where $b \in \mathbb{R}^n$ is a given cost vector, the set $\mathcal{X}$ represents the deterministic constraints on the variables x, and $\mathbb{P}(\mathcal{B}(x)) \geq 1 - \epsilon$ represents the restriction that the probability of event $\mathcal{B}(x)$ must be at least $1 - \epsilon$. There are three sources of difficulty for this class of problems. First, for a given x, computing $\mathbb{P}(\mathcal{B}(x))$ exactly is hard in general, because it involves multi-dimensional integrals. Second, the feasible region of chance-constrained programs is non-convex for general probability distributions. Finally, due to the combinatorial nature of the decisions, the search space is very large.

In the CCP literature, the first challenge of evaluating the probability of an event, $\mathbb{P}(\mathcal{B}(x))$, is generally overcome by sampling from the true distribution [77, 83, 13, 67, 55]. This creates an approximation of the chance constraint, which can be evaluated for the given

samples. In contrast, in this chapter, we assume that there exists an efficient oracle, which provides an exact value of $\mathbb{P}(\mathcal{B}(x))$ for a given x. We give a delayed cut generation algorithm to solve problem (3.1) exactly using the true distribution, instead of sampling from the true distribution. We show that probabilistic partial set covering problems (PPSC) under certain distributions admit an efficient probability oracle. Using this class of problems in our computational study, we demonstrate the effectiveness of the proposed algorithm for small-size problems. However, due to the exponential decision space, convergence may be slow for larger problems. We observe that the explicit (linear) formulation of a chance constraint using a sampling-based approach may be more amenable to exploiting the special structure of the underlying problem. Hence the sampling-based approach may be more effective in solving larger problems. However, the solutions to the sample approximation problem may not satisfy the chance constraint under the true distribution if not enough samples are used. On the other hand, the solution methods may be slow for large sample sizes. For such a sampling-based approach, we propose a method that utilizes the oracle to check and correct the feasibility of the approximate solution by adding *globally valid* feasibility cuts to the sample approximation problem. Note that for structured problems, existing sampling-based approaches also add feasibility cuts exploiting the structure of the problem (see, e.g., [93, 109, 52]). However, such cuts for the sample approximation problem are valid based on the scenarios generated, but they may not be globally valid with respect to the original problem under the true distribution.)

We handle the second difficulty (non-convexity of the feasible region) by expressing the feasibility condition using linear constraints. For the sample approximation problem, such a linear reformulation using additional binary variables is well known [56]. For the general case (without sampling), we consider a reformulation with exponentially many linear inequalities. We solve this formulation using a delayed cut generation algorithm, which starts with a subset of the inequalities, and adds the violated inequalities as needed to cut off the infeasible solutions until a feasible and optimal solution is found. In addition, for a special case, we show that there exists a compact (polynomial-size) mixed integer linear program (MIP) that

solves the problem without the need for sampling. To handle the third difficulty, we show that we can use the properties of the oracle to obtain stronger inequalities to represent the feasibility conditions, which, in turn, reduce the search space of problem (3.1) significantly.

Under certain conditions, if the finite dimensional distributions of the uncertain parameters are log-concave probability measures, then the continuous relaxation of the chance constraint is convex [76]. van Ackooij et al. [95] consider such convex chance-constrained combinatorial optimization problems, where the objective function is non-differentiable. The authors use a sampling (scenario)-based approach and introduce additional binary variables to represent whether the chance constraint is satisfied under each scenario. They propose a Benders decomposition algorithm using combinatorial Benders (no-good) cuts, where they use an inexact oracle to approximate the non-differentiable objective value. In contrast, we assume that the objective function is smooth (linear) and use an exact oracle for evaluating the *non-convex* chance constraint. In another line of work, van Ackooij and Sagastizábal [96] consider CCPs, where the chance constraint is convex but hard to evaluate exactly, and the additional constraints on the decision variables form a convex set (as a result, the decision variables are continuous). The authors give a non-smooth optimization (bundle) method that uses an inexact oracle to evaluate the chance constraint to find an approximate solution. In contrast, in our problem (3.1), we do not assume convexity of the chance constraint $\mathbb{P}(\mathcal{B}(x)) \geq 1 - \epsilon$, or the continuity of the decision variables, the binary restrictions on the decision variables form a non-convex set.

We demonstrate our proposed methods on a probabilistic partial set covering problem (PPSC) introduced in [110] for bipartite social networks. Given a collection of n subsets of m items, a deterministic set covering problem aims to choose subsets among the collection at a minimum cost, such that each item is covered by at least one chosen subset. In the probabilistic version of this problem we consider, it is assumed that when a subset is chosen, there is uncertainty in which items in the subset are actually covered. Given a fixed target $\tau \leq m$, the probabilistic *partial* set covering problem (PPSC) aims to find the minimum cost selection of subsets, which cover at least the target number of items, τ, with probability

$1 - \epsilon$. (Note that for $\tau = m$, this problem is equivalent to probabilistic set covering.) Under certain distributions of the random variables, there exists a polynomial-time oracle to check the feasibility of a given selection of subsets. Using this oracle, we give an exact delayed cut generation algorithm to find the optimal solution. This is equivalent to solving an exponential-sized integer linear program, where an efficient separation algorithm is available. In addition, we show that for a special case of interest, the oracle is formulable and it can be incorporated into the optimization model, which results in a polynomial-sized mixed-integer linear program. While both of these approaches find optimal solutions to moderate-size problems, the solution times grow exponentially as the problem size increases. In such cases, we develop a modified sampling-based method for PPSC that is able to exploit the special structure of the problem, namely the submodularity. We derive a new class of valid inequalities for PPSC that subsumes the submodular inequalities of Nemhauser and Wolsey [66], and provide conditions under which the proposed inequalities are facet defining. We observe that the modified sampling-based method is highly effective when combined with the probability oracle to obtain feasible solutions of good quality. The literature review on probabilistic set covering problems and further discussions on alternative approaches are given in the corresponding section (Section 3.3).

We summarize our contributions and give an outline of this chapter as follows. In Section 3.2, we introduce the concept of a probability oracle for a class of combinatorial CCPs and propose a general method to solve such CCPs that uses the concept of no-good cuts. We strengthen the no-good cuts by using the monotonicity of the probability function and the availability of the oracle. In Section 3.3, we use a class of $\mathcal{NP}$-hard problems (PPSC) to demonstrate the proposed method. In addition, we show that we can solve PPSC by using a compact deterministic MIP under a special case. Furthermore, we propose a modified sampling-based method for PPSC that utilizes its submodular substructure. We introduce a new class of facet-defining inequalities for this substructure of PPSC. In addition, we show that an efficient oracle can be a useful tool for checking and correcting the feasibility of a solution given by a sampling-based approach. Furthermore, we propose a modified sampling-

based method for PPSC that provides a high-quality feasible solution to the true problem. We introduce a new class of facet-defining inequalities for the submodular substructure of PPSC that subsumes the known submodular inequalities. We show that we can solve the sample approximation problem of PPSC by using a compact deterministic MIP under a special case. In Section 3.4, we report the computational results with these alternative approaches.

3.2 Chance-Constrained Combinatorial Optimization with a Probability Oracle

Suppose that we have an oracle $\mathcal{A}(x)$, which computes $\mathbb{P}(\mathcal{B}(x))$ exactly for a given x in polynomial time. We reformulate problem (3.1) as

$$\min\{b^\top x : \mathcal{A}(x) \geq 1-\epsilon, x \in \mathcal{X} \cap \mathbb{B}^n\}. \tag{3.2}$$

In general, it is hard to compute $\mathcal{A}(x)$, it involves high dimensional integrals, or in some cases, it is a black box evaluated by simulation methods. In addition, constraint $\mathcal{A}(x) \geq 1-\epsilon$ is highly non-convex, in general. In this section, we propose a general delayed cut generation approach to solve formulation (3.2) when an exact oracle for $\mathcal{A}(x)$ exists.

Here we address a general approach to solve formulation (3.2) exactly. The algorithm works by solving a relaxed problem, and cutting off infeasible solutions iteratively until we find an optimal solution. Consider the generic relaxed master problem (RMP) of formulation (3.2) as

$$\min\{b^\top x : x \in \mathcal{C} \cap \mathcal{X} \cap \mathbb{B}^n\}, \tag{3.3}$$

where $\mathcal{C}$ is a set of feasibility cuts added until the current iteration. We describe a delayed constraint generation approach with the probability oracle in Algorithm 4. To solve formulation (3.2), Algorithm 4 starts with a subset of feasibility cuts in $\mathcal{C}$ (could be empty) in RMP (3.3). At each iteration (Lines 2-9), solving RMP (3.3) provides an incumbent solution $\bar{x}$ (Line 3). Then the oracle $\mathcal{A}(\bar{x})$ is used as a separation routine to check the feasibility of $\bar{x}$. Note that $\mathcal{A}(\bar{x}) \geq 1-\epsilon$ in Line 4 is the feasibility condition of $\bar{x}$. If $\bar{x}$ is feasible,

then we break the loop and declare the optimal solution as $\bar{x}$ (Lines 5 and 11); otherwise, a subroutine FeasibilityCut$(\bar{x}, \kappa, \mathcal{C})$ is called with input $\bar{x}$ and optional parameters κ. The subroutine adds a feasibility cut to the current set $\mathcal{C}$ (Line 8) to cut off $\bar{x}$ in further iterations. We specify this subroutine and the corresponding cuts next.

Algorithm 4: An Exact Delayed Constraint Generation Algorithm with a Probability Oracle

```
1  Start with an initial set of feasibility cuts in C (could be empty);
2  while True do
3  |  Solve master problem (3.3), and obtain an incumbent solution x̄ ;
4  |  if A(x̄) ≥ 1 − ε then
5  |  |  break;
6  |  end
7  |  else
8  |  |  Call FeasibilityCut(x̄, κ, C) ;
9  |  end
10 end
11 Output x̄ as an optimal solution.
```

Let $V_1 := \{1, \ldots, n\}$. Given an incumbent solution $\bar{x}$ such that $\mathcal{A}(\bar{x}) < 1 - \epsilon$, let $J_1 = \{i \in V_1 | \bar{x}_i = 1\}$ and $J_0 = \{j \in V_1 | \bar{x}_j = 0\}$. A class of feasibility cuts, commonly known as no-good cuts, is given by

$$\sum_{i \in J_1} (1 - x_i) + \sum_{j \in J_0} x_j \geq 1, \tag{3.4}$$

which ensures that if $\bar{x}$ is infeasible, then at least one component in $\bar{x}$ must be changed. Laporte and Louveaux [44] provide a review of inequality (3.4) for two-stage stochastic programs where the first-stage problem is pure binary and second-stage problem is mixed-integer.

In the next proposition, we observe that if $\mathbb{P}(\mathcal{B}(x))$ is monotonically increasing in x for problem (3.1), then a stronger inequality is valid for formulation (3.2). Throughout, we let $\mathbf{e}_j$ be a unit vector of dimension n whose jth component is 1.

Proposition 4. *Suppose that $\mathbb{P}(\mathcal{B}(x))$ is a monotonically increasing function in x. Given a vector $\bar{x}$ with $J_0 = \{i \in V_1 : \bar{x}_i = 0\}$ and $J_1 = V_1 \setminus J_0$ and $\mathbb{P}(\mathcal{B}(\bar{x})) < 1-\epsilon$, let $\kappa(J_0) < |J_0|$ be a positive integer such that $\forall \mathcal{K} \subseteq J_0$ with $|\mathcal{K}| = \kappa(J_0)-1$, we have $\mathbb{P}(\mathcal{B}(\bar{x}+\sum_{j\in\mathcal{K}} \mathbf{e}_j)) < 1-\epsilon$.*

(i) The inequality

$$\sum_{j\in J_0} x_j \geq \kappa(J_0), \tag{3.5}$$

is valid for formulation (3.2).

(ii) Inequality (3.5) *is stronger than inequality* (3.4) *for the same choice of J_0.*

Proof. (i) Let $\bar{x}' \neq \bar{x}$ denote another vector, where $\bar{J}_1' = \{i \in V_1 | \bar{x}_i' = 1\}$ and $\bar{J}_1' \subset J_1$. Since $\mathbb{P}(\mathcal{B}(x))$ is a monotonically increasing function, $\bar{J}_1' \subset J_1$ implies that $\mathbb{P}(\mathcal{B}(\bar{x}')) \leq \mathbb{P}(\mathcal{B}(\bar{x})) < 1-\epsilon$, so $\bar{x}'$ is also infeasible for formulation (3.2). Recall that for all $\mathcal{K} \subseteq J_0$, where $|\mathcal{K}| \leq \kappa(J_0)-1$, we have $P(\mathcal{B}(\bar{x}+\sum_{j\in\mathcal{K}} \mathbf{e}_j)) < 1-\epsilon$. Then, for a feasible solution x', with $J_1' = \{i \in V_1 | x_i' = 1\} \supseteq J_1$ and $\mathbb{P}(\mathcal{B}(x')) \geq 1-\epsilon$, we must have $|J_1' \setminus J_1| \geq \kappa(J_0)$. Note that $J_1' \setminus J_1 \subseteq J_0$. Hence, $\sum_{j\in J_0} x_j \geq \sum_{j\in J_1'\setminus J_1} x_j \geq \kappa(J_0)$, which proves the claim.

(ii) Note that for the same choice of J_0, we have $\sum_{i\in J_1}(1-x_i) + \sum_{j\in J_0} x_j \geq \sum_{j\in J_0} x_j \geq \kappa(J_0) \geq 1$, where the first inequality follows because $\sum_{i\in J_1}(1-x_i) \geq 0$. The result then follows.

□

In light of Proposition 4, we specify the subroutine FeasibilityCut$(\bar{x}, \kappa, \mathcal{C})$ in Algorithm 5 for the case that $\mathcal{A}(\bar{x})$ is monotone. If $\mathcal{A}(\bar{x})$ is not monotone, then inequality (3.5) should be replaced with inequality (3.4) in Algorithm 5. In this subroutine, we seek inequalities (3.5) with $\kappa(J_0) \leq \kappa$, given the input parameter κ. If $\kappa = 2$, then we check if there exists

some $j \in J_0$ for which $\mathcal{A}(\bar{x} + \mathbf{e}_j) \geq 1 - \epsilon$. In other words, we check if there exists a feasible solution after letting $x_j = 1$ for some $j \in J_0$. If so, we let $\kappa(J_0) = 1$ for the inequality to be valid. If such a j does not exist, then this implies that complementing one variable that is in J_0 is not sufficient to obtain a feasible solution. In this case, we let $\kappa(J_0) = 2$ in inequality (3.5). Note that higher values of κ than 2 will require more computational effort, so we only consider $\kappa = 2$ in Algorithm 5.

Algorithm 5: Subroutine FeasibilityCut$(\bar{x}, \kappa, \mathcal{C})$

```
for j ∈ J_0 do
    if A(x̄ + e_j) ≥ 1 − ε then
        Add inequality (3.5) with κ(J_0) = 1 to C;
        BoundIncrease = 0;
        break;
    end
end
if κ = 2 then
    Add inequality (3.5) with κ(J_0) = 2 to C;
end
```

Next, we demonstrate the proposed algorithm on a class of probabilistic set covering problems, and provide alternative solution approaches utilizing an efficient probability oracle.

3.3 An Application: A Probabilistic Partial Set Covering Problem

In this section, we study a probabilistic partial set covering problem (PPSC) as an application of problem (3.1). First, we describe the deterministic set covering problem. The deterministic set covering problem is a fundamental combinatorial optimization problem that arises in many applications, such as facility selection, scheduling, and manufacturing. We refer the reader to [9] for a review of the various applications of the set covering problem. For example,

in the facility selection problem, there are n facilities given by the set V_1 and m customers given by the set V_2. Suppose that facility j covers (satisfies the demand of) customer i if the travel time between the facility and the customer is within a pre-specified time limit. In this case, we can form a set S_j as those customers who are within the acceptable time limit away from facility j. Given the cost of building facility j, b_j, the set covering problem aims to find the minimum cost selection of facilities that cover all customers.

More formally, given a set of items $V_2 := \{1, \ldots, m\}$ and a collection of n subsets $S_j \subseteq V_2, j \in V_1 := \{1, \ldots, n\}$ such that $\cup_{j=1}^{n} S_j = V_2$, the deterministic set covering problem is defined as

$$\min \quad \sum_{j \in V_1} b_j x_j \tag{3.6a}$$

$$\text{s.t.} \quad \sum_{j \in V_1} t_{ij} x_j \geq h_i \qquad \forall i \in V_2 \tag{3.6b}$$

$$x \in \mathbb{B}^n, \tag{3.6c}$$

where b_j is the objective coefficient of x_j, $h_i = 1$ for all $i \in V_2$, and $t_{ij} = 1$ if $i \in S_j$; otherwise, $t_{ij} = 0$ for all $i \in V_2 \setminus S_j, j \in V_1$. Karp [36] proves that the set covering problem is $\mathcal{NP}$-hard.

Probabilistic set covering extends this problem to the probabilistic setting to capture uncertain travel times. In the stochastic variant of the set covering problem we consider, the chance constraint ensures a high quality of service as measured by serving a target number $\tau \leq m$ of the customers within preferred time limits with high probability. Different variants of the probabilistic set covering problem have been considered in the literature, wherein constraint (3.6b) is replaced with a chance constraint when either the constraint coefficients t_{ij} or the right-hand side h_i is assumed to be random for $i \in V_2, j \in V_1$. Beraldi and Ruszczyński [14] and Saxena et al. [86] study the uncertainty in the right-hand side of constraint (3.6b), in other words h_i is assumed to be a binary random variable. Fischetti and Monaci [30] and Ahmed and Papageorgiou [5] study the uncertainty with the randomness in the coefficients of constraint (3.6b), i.e., t_{ij} is a binary random variable indicating whether set j covers item i. They consider *individual* chance constraints that ensure that each item

is covered with a certain probability. In this chapter, we focus on the uncertainty in t_{ij} for all $i \in V_2$ and $j \in V_1$. In addition, we study a version of PPSC such that the probability that the selected subsets cover a given number τ of items in V_2 is at least $1 - \epsilon$, which we describe next. (Note that when $\tau = m$, our model considers the *joint* probability of covering *all* customers.)

Let $\sigma(x)$ be a random variable representing the number of covered items in V_2 for a given x. For example, in the facility selection problem with random travel times, $\sigma(x)$ represents the number of customers for whom the travel time from a selected facility $j \in V_1$ (with $x_j = 1$) is within the pre-specified time limit. Suppose that we are given the cost b_i of each set $i \in V_1$, a target τ of the number of covered items in V_2, and a risk level $\epsilon \in [0, 1]$. The variant of the probabilistic set covering model we consider is

$$\min \quad \sum_{i \in V_1} b_i x_i \tag{3.7a}$$

$$\text{s.t.} \quad \mathbb{P}(\sigma(x) \geq \tau) \geq 1 - \epsilon \tag{3.7b}$$

$$x \in \mathbb{B}^n, \tag{3.7c}$$

where $\sigma(x) \geq \tau$ is the desired covering event, $\mathcal{B}(x)$, for a given x. For $\tau = m$, this problem is equivalent to a probabilistic set covering model, where each node must be covered. However, because we allow $\tau \leq m$, we refer to this model as probabilistic partial set covering. Note that $\sigma(x)$ is a submodular function [38]. Our goal is to minimize the total cost of the sets selected from V_1 while guaranteeing a certain degree of coverage of the items in V_2.

We represent the partial set covering problem on a bipartite graph $G = (V_1 \cup V_2, E)$. There are two groups of nodes V_1 and V_2 in G, where all arcs in E are from V_1 to V_2. Node $i \in V_1$ represents set S_i and nodes in $j \in V_2$ represent the items. There exists an arc $(i, j) \in E$ representing the covering relationship if $j \in S_i$ for $i \in V_1$. In probabilistic set covering, the covering relationship is stochastic, in other words an item i may not be covered by the subset S_j, $j \in V_1$, even though $i \in S_j$. In this chapter, we consider probabilistic partial set covering problems (PPSC) under two probability distributions:

Probabilistic Partial Set Covering with Independent Probability Coverage: In this model, each node j has an independent probability a_{ij} of being covered by node i for $j \in S_i$.

Probabilistic Partial Set Covering with Linear Thresholds: In the linear threshold model of Kempe et al. [38], each arc $(i,j) \in E$ has a deterministic weight $0 \leq a_{ij} \leq 1$, such that for all nodes $j \in V_2$, $\sum_{i:(i,j)\in E} a_{ij} \leq 1$. In addition, each node $j \in V_2$ selects a threshold $\nu_j \in [0,1]$ *uniformly at random.* A node $j \in V_2$ is covered if sum of the weights of its selected neighbors $i \in V_1$ is above its threshold, i.e., $\sum_{i:(i,j)\in E} a_{ij}x_i \geq \nu_j$.

The probabilistic models we consider may be seen as chance-constrained extensions of the independent cascade and linear threshold models in social networks proposed by Kempe et al. [38], applied to bipartite graphs. Zhang et al. [110] first proposed a model to find the minimum number of individuals to influence a target number of people in social networks with a probability guarantee, where one individual has an independent probability of influencing another individual. Note that if the social network is a bipartite graph, the question proposed by Zhang et al. [110] can be formulated as PPSC. Zhang et al. [110] describe a polynomial time algorithm to compute $\mathbb{P}(\sigma(x) \geq \tau)$ exactly for bipartite graphs under certain probability distributions for a given x. They propose a greedy heuristic to obtain a solution to PPSC, which has an $O(m+n)$ multiplicative error and an $O(\sqrt{m+n})$ additive error on the quality of the solution for the case that $b_i = 1$ for all $i \in V_1$, and no performance guarantee on the quality of the solution for the general cost case. In contrast, we give an exact algorithm to find an optimal x. Next, we review an efficient oracle for PPSC under the distributions of interest for both versions of PPSC.

3.3.1 An Oracle

Let $P(x,i)$ be the probability that a given solution x covers node $i \in V_2$. For the linear threshold model, $P(x,i) = \sum_{j\in V_1} a_{j,i}x_j$, and for the independent probability coverage model, $P(x,i) = 1 - \prod_{j\in V_1}(1-a_{j,i}x_j)$. For a given x, the probability of covering exactly k nodes in V_2 out of a total of $|V_2| = m$ is represented as $\mathbb{P}(\sigma(x) = k)$, and $\mathbb{P}(\sigma(x) \geq \tau) = \sum_{k=\tau}^{m} \mathbb{P}(\sigma(x) =$

k). Note that $\mathbb{P}(\sigma(x) = k)$ is equal to the probability mass function of the Poisson binomial distribution [35, 84, 102], which is the discrete probability distribution of k successes in m Bernoulli trials, where each Bernoulli trial has a unique success probability. Let $\mathbb{P}_i(\sigma(x) = j)$ denote the probability of having j covered nodes in $V_2 \setminus \{i\}$ for a given x for any $j = 0, \ldots, k$. Samuels [84] provides a formula to obtain the value of probability mass function of the Poisson binomial distribution:

$$\mathbb{P}(\sigma(x) = j) = P(x, i) \times \mathbb{P}_i(\sigma(x) = j - 1) + (1 - P(x, i)) \times \mathbb{P}_i(\sigma(x) = j). \tag{3.8}$$

Next, we describe a dynamic program (DP) to compute $\mathbb{P}(\sigma(x) \geq \tau)$ exactly [11, 110]. Let $V^i = \{1, \ldots, i\} \in V_2$ be the set of the first i nodes of V_2. Also let $A(x, i, j)$ represent the probability that the selection x covers j nodes among V^i for $0 \leq j \leq i, i \in V_2$. The DP recursion for $A(x, i, j)$ for $1 \leq j \leq i, i \in V_2$ is formulated as

$$A(x, i, j) = \begin{cases} A(x, i-1, j) \times (1 - P(x, i)), & j = 0 \\ A(x, i-1, j) \times (1 - P(x, i)) + A(x, i-1, j-1) \times P(x, i), & 0 < j < i \\ A(x, i-1, j-1) \times P(x, i), & j = i, \end{cases}$$

where the boundary condition is $A(x, 0, 0) = 1$. The goal function $\sum_{j=\tau}^{m} A(x, m, j)$ calculates the probability that the number of covered nodes is at least the target τ for a given x. In other words, $\mathcal{A}(x) = \mathbb{P}(\sigma(x) \geq \tau) = \sum_{j=\tau}^{m} A(x, m, j)$. For a given x, the running time of this DP is $\mathcal{O}(nm + m^2)$, because obtaining $P(x, j)$ for all $j \in V_2$ is $\mathcal{O}(nm)$, and computing the recursion is $\mathcal{O}(m^2)$.

Next, we show that $\mathcal{A}(x)$ is a monotone increasing function in x. We use the following lemma as a tool to prove the property of $\mathbb{P}(\sigma(x) \geq \tau)$.

Lemma 1. *[Lemma 2.12 in [97]] Let R_1 and R_2 be two random variables defined on two different probability spaces. The random variables $\hat{R}_1$ and $\hat{R}_2$ are a coupling of R_1 and R_2 when $\hat{R}_1$ and $\hat{R}_2$ are defined in the same probability space. In addition, the marginal distribution of $\hat{R}_1$ is the same as R_1, and the marginal distribution of $\hat{R}_2$ is the same as R_2.*

Given $\gamma \in \mathbb{R}$, $\mathbb{P}(R_1 \geq \gamma) \leq \mathbb{P}(R_2 \geq \gamma)$ if and only if there exists a coupling $\hat{R}_1$ and $\hat{R}_2$ of R_1 and R_2 such that $\mathbb{P}(\hat{R}_1 \leq \hat{R}_2) = 1$.

Proposition 5. *Given τ, $\mathbb{P}(\sigma(x) \geq \tau)$ is a monotonically increasing function in x for the probabilistic partial set covering problem under the independent probability coverage and the linear threshold models.*

Proof. Suppose that we have two binary vectors, x' and x''. Let X' and X'' be the support of the vectors x' and x'', and suppose that $X' \subseteq X''$. Recall that for the linear threshold model, $P(x,i) = \sum_{j \in V_1} a_{j,i} x_j$, and for the independent probability coverage model, $P(x,i) = 1 - \prod_{j \in V_1}(1 - a_{j,i}x_j)$. Since $P(x,i)$ is a monotone increasing function in x, $P(x',i) \leq P(x'',i)$ for all $i = 1, \ldots, m$. We construct two random variables $\hat{\sigma}(x')$ and $\hat{\sigma}(x'')$, which is a coupling of $\sigma(x')$ and $\sigma(x'')$. We apply the technique shown in [75] (Chapter 10, Example 1) to generate $\hat{\sigma}(x')$ and $\hat{\sigma}(x'')$. Let U_i be an independent random variable with uniform distribution in $[0,1]$ for all $i = 1, \ldots, m$. Let $\hat{\sigma}_i(x')$ be a random variable for all $i = 1, \ldots, m$ such that $\hat{\sigma}_i(x') = 1$ if $U_i \leq P(x',i)$, and 0 otherwise. Let $\hat{\sigma}_i(x'')$ be a random variable for all $i = 1, \ldots, m$ such that $\hat{\sigma}_i(x'') = 1$ if $U_i \leq P(x'',i)$, and 0 otherwise. We set $\hat{\sigma}(x') = \sum_{i=1}^{m} \sigma_i(x')$ and $\hat{\sigma}(x'') = \sum_{i=1}^{m} \sigma_i(x'')$. Since $P(x',i) \leq P(x'',i)$, $\hat{\sigma}_i(x') \leq \hat{\sigma}_i(x'')$ for all $i = 1, \ldots, m$. Then, $\hat{\sigma}(x') \leq \hat{\sigma}(x'')$. From Lemma 1, if $\mathbb{P}(\hat{\sigma}(x') \leq \hat{\sigma}(x'')) = 1$, then $\mathbb{P}(\sigma(x') \geq \tau) \leq \mathbb{P}(\sigma(x'') \geq \tau)$. This completes the proof. □

Proposition 5 proves the intuitive result that if more nodes from V_1 are selected, then we have a higher chance to cover more nodes from V_2. Consequently, Proposition 5 allows us to use the stronger inequality (3.5) in Algorithm 4. In the following subsections, we employ the DP described in this section first to reformulate the problem using a compact mathematical model, and then in Algorithm 5 as the oracle $\mathcal{A}(x)$ for PPSC.

3.3.2 A Compact MIP for PPSC with a Probability Oracle

Using the DP representation of the oracle $\mathcal{A}(x) = \mathbb{P}(\sigma(x) \geq \tau)$, PPSC problem (3.7) can be reformulated as a compact mathematical program

$$\min \quad \sum_{i \in V_1} b_i x_i \tag{3.9a}$$

$$\text{s.t.} \quad \bar{A}_{0,0} = 1 \tag{3.9b}$$

$$\bar{A}_{i,j} = \bar{A}_{i-1,j}(1 - P(x,i)), \quad i = 1, \ldots, m; j = 0 \tag{3.9c}$$

$$\bar{A}_{i,j} = \bar{A}_{i-1,j}(1 - P(x,i)) + \bar{A}_{i-1,j-1}P(x,i), \tag{3.9d}$$

$$i = 1, \ldots, m; 0 < j < i$$

$$\bar{A}_{i,j} = \bar{A}_{i-1,j-1}P(x,i), \quad i = 1, \ldots, m; j = i \tag{3.9e}$$

$$\sum_{j=\tau}^{m} \bar{A}_{m,j} \geq 1 - \epsilon \tag{3.9f}$$

$$x \in \mathbb{B}^n \tag{3.9g}$$

$$\bar{A}_{i,j} \in \mathbb{R}_+, \quad 0 \leq j \leq i \leq m, \tag{3.9h}$$

where $\bar{A}_{i,j}$ is a decision variable representing $A(x,i,j)$ defined in the DP formulation (we drop the dependence on x for ease of notation). Constraint (3.9b) is the boundary condition of the DP, constraints (3.9c)–(3.9e) are the DP recursive functions, and constraint (3.9f) is the goal function. Note that $P(x,i)$ is a function of the decision vector x. Hence, formulation (3.9) is a mixed-integer *nonlinear* program due to the constraints (3.9c)-(3.9e). Depending on the complexity of the function $P(x,i)$, this formulation may be difficult to solve. However, for a special case of PPSC, namely the linear threshold model, the nonlinear programming model can be reformulated as a *linear* mixed-integer program (MIP). Recall that for the linear threshold model, $P(x,i) = \sum_{u \in V_1} a_{u,i} x_u$.

Hence, the term $\bar{A}_{i,j}x_j$ appearing in (3.9c)-(3.9e) is a bilinear term, which can be linearized [60, 2]. To this end, we introduce the additional variables $\gamma_{u,i,j} = \bar{A}_{i,j}x_u$ for $u \in V_1, i \in V_2, 0 \leq$

$j \leq i$, and obtain an equivalent linear MIP

$$\min \quad \sum_{i \in V_1} b_i x_i \tag{3.10a}$$

$$\text{s.t.} \quad (3.9b), (3.9f) - (3.9h) \tag{3.10b}$$

$$\bar{A}_{i,j} = \bar{A}_{i-1,j} - \sum_{u \in V_1} a_{u,i} \gamma_{u,i-1,j}, \quad i = 1, \ldots, m; j = 0 \tag{3.10c}$$

$$\bar{A}_{i,j} = \bar{A}_{i-1,j} - \sum_{u \in V_1} a_{u,i} \gamma_{u,i-1,j} + \sum_{u \in V_1} a_{u,i} \gamma_{u,i-1,j-1}, \tag{3.10d}$$

$$i = 1, \ldots, m; 0 < j < i$$

$$\bar{A}_{i,j} = \sum_{u \in V_1} a_{u,i} \gamma_{u,i-1,j-1}, \quad i = 1, \ldots, m; j = i \tag{3.10e}$$

$$\gamma_{u,i,j} \leq x_u, \quad i = 0, \ldots, m; j = 0, \ldots, i; u \in V_1 \tag{3.10f}$$

$$\gamma_{u,i,j} \leq \bar{A}_{i,j}, \quad i = 0, \ldots, m; j = 0, \ldots, i; u \in V_1 \tag{3.10g}$$

$$\gamma_{u,i,j} \geq \bar{A}_{i,j} - (1 - x_u), \quad i = 0, \ldots, m; j = 0, \ldots, i; u \in V_1 \tag{3.10h}$$

$$\gamma_{u,i,j} \in \mathbb{R}_+, \quad i = 0, \ldots, m; j = 0, \ldots, i; u \in V_1. \tag{3.10i}$$

The DP recursion is represented in constraints (3.10c)-(3.10e). Constraints (3.10f)-(3.10i) are the McCormick linearization constraints to ensure that if $x_u = 0$, then $\gamma_{u,i,j} = 0$, and if $x_u = 1$, then $\gamma_{u,i,j} = \bar{A}_{i,j}$. As a result, in this special case, the oracle is a formulable function, and the linear threshold model can be solved exactly with the compact MIP (3.10). Alternatively, Algorithm 4 can be used to solve the exponential formulation of (3.7) by delayed constraint generation. We close this subsection by noting that, for the case of the independent probability coverage model, the mixed-integer nonlinear program (3.9) is multilinear due to the terms $\bar{A}_{i-1,j-1}(1 - \prod_{u \in V_1}(1 - a_{u,i} x_u))$. While such multilinear terms can also be linearized with successive application of the McCormick linearization, the resulting formulations are large scale and they suffer from weak LP relaxations. Therefore, we do not pursue such formulations for the independent probability coverage model in our computational study.

3.3.3 A General Decomposition Approach for PPSC with a Probability Oracle

Algorithm 4 can be used to solve formulation (3.7) exactly. To update $\kappa(J_0)$ in Algorithm 5 for $\kappa = 2$ more efficiently, we utilize the DP structure of $\mathcal{A}(\bar{x})$. Note that when we obtain a solution $\bar{x}$, with an associated J_0, we first calculate $\mathcal{A}(\bar{x})$ using the DP, which requires the calculation of $P(\bar{x}, i), \forall i \in V_2$. Then, to calculate inequalities with $\kappa(J_0) = 2$, we need to calculate $\mathcal{A}(\bar{x} + \mathbf{e}_j)$ for each $j \in J_0$. If we calculate $P(\bar{x} + \mathbf{e}_j, i), \forall i \in V_2$ for each $j \in J_0$ from scratch, the time complexity is $\mathcal{O}(nm)$ for the independent probability coverage and the linear threshold models. However, given that we have just calculated (and stored) all $P(\bar{x}, i)$ values, the time complexity of updating $P(\bar{x}, i)$ to $P(\bar{x} + \mathbf{e}_j, i), \forall i \in V_2, j \in J_0$ is $\mathcal{O}(m)$.

As we will show in our computational study, when the number of decision variables is large, Algorithm 4 exhibits slow convergence, even if there exists an efficient probability oracle. In this case, Algorithm 4 may check a large (worst case exponential) number of incumbent solutions $\bar{x}$ to obtain the optimal solution. In the next section, we consider a sampling-based approach to find approximate solutions to PPSC. The sampling-based approach enables us to use the problem structure to expedite the convergence to a solution, but the optimal solution to the sample approximation problem may not be feasible with respect to the true distribution. In this case, the probability oracle is used as a detector to check and correct the infeasibility of the solution given by the sampling-based approach.

3.3.4 A Sampling-Based Approach for PPSC with a Probability Oracle

Using sampling-based methods, we can approximately represent the uncertainty with a finite number of possible outcomes (known as scenarios). This, in turn, allows us to rewrite the non-convex chance constraint as linear inequalities with big-M coefficients, if the desirable event $\mathcal{B}(x)$ has a linear representation. Such a formulation is known as the deterministic equivalent formulation. Luedtke et al.; Küçükyavuz; Abdi and Fukasawa; Zhao et al. and Liu et al. [56, 42, 1, 111, 51] introduce strong valid inequalities for the deterministic equivalent

formulation of linear chance constraints under right-hand side uncertainty. Ruszczyński; Beraldi and Bruni; Lejeune; Luedtke and Liu et al. [83, 13, 46, 54, 53] study general CCPs with the randomness in the coefficient (technology) matrix (including two-stage CCPs), and propose solution methods for the sampling-based approach. In another line of work, Song et al. [93] consider a special case of combinatorial chance-constrained programs, namely the chance-constrained packing problems under finite discrete distributions, and give a delayed constraint generation algorithm using the so-called probabilistic cover and pack inequalities valid for the chance-constrained binary packing problems.

In this section, we consider related sampling-based reformulations of PPSC. First, we describe how we sample from the true distribution to obtain a set of scenarios (sample paths) Ω for PPSC (see [38, 105] for a detailed description). For the case of the independent probability coverage model, we generate a scenario by tossing biased coins for each arc $(i, j) \in E$ with associated probability a_{ij}. The coin tosses reveal if node $j \in V_2$ is covered by node $i \in V_2$ in which case we refer to arc $(i, j) \in E$ as a live arc. For each sample (scenario) $\omega \in \Omega$, with a probability of occurrence p_ω, a so-called *live-arc graph* $G_\omega = (V_1 \cup V_2, E_\omega)$ is constructed, where E_ω is the set of live arcs under scenario ω. We refer the reader to Kempe et al. [38] for a scenario generation method for the linear threshold model, which results in live-arc graphs G_ω for each $\omega \in \Omega$. It is important to note that in the live-arc graphs of linear threshold models, each node in V_2 has at most one incoming arc. Let $t_{ij}^\omega = 1$ if arc $(i, j) \in E_\omega$ for $\omega \in \Omega$, and $t_{ij}^\omega = 0$ otherwise.

Given a set of scenarios Ω and a target level τ, we can reformulate the submodular formulation (3.7) of PPSC as a two-stage chance-constrained program under a finite discrete distribution. In the first stage, the nodes from set V_1 are selected by the decision vector x. Then the uncertainty unfolds, and live arcs are realized. The second-stage problem for each scenario determines the number of nodes in V_2 covered by the nodes in V_1 selected in the first stage. Let $y_i^\omega = 1$ if node $i \in V_2$ is covered by the node selection x under scenario $\omega \in \Omega$.

Then a deterministic equivalent formulation for PPSC is

$$\min \quad \sum_{j \in V_1} b_j x_j \tag{3.11a}$$

$$\text{s.t.} \quad \sum_{j \in V_1} t_{ij}^{\omega} x_j \geq y_i^{\omega} \qquad \forall i \in V_2, \forall \omega \in \Omega \tag{3.11b}$$

$$\sum_{i \in V_2} y_i^{\omega} \geq \tau z_{\omega} \qquad \forall \omega \in \Omega \tag{3.11c}$$

$$\sum_{\omega \in \Omega} p_{\omega} z_{\omega} \geq 1 - \epsilon \tag{3.11d}$$

$$x \in \mathbb{B}^n, y \in \mathbb{B}^{m \times |\Omega|}, z \in \mathbb{B}^{|\Omega|}, \tag{3.11e}$$

where constraints (3.11b) ensure that $y_i^{\omega} = 1$ if node $i \in V_2$ is covered by the node selection x under scenario $\omega \in \Omega$, and constraints (3.11c) ensure that if $z_{\omega} = 1$, then $\sum_{i \in V_2} y_i^{\omega} \geq \tau$ for all $\omega \in \Omega$. Constraint (3.11d) ensures that the probability that τ items are covered in V_2 is at least $1 - \epsilon$. Formulation (3.11) is a very large-scale MIP that continues to challenge the state-of-the-art optimization solvers. Instead, delayed constraint generation methods akin to Benders decomposition method [54, 53] are known to be computationally more effective for such problems. Because the second-stage problem is concerned with feasibility only, the decomposition algorithm proposed in [54] is applicable to this formulation (Liu et al. [53] also consider the second-stage objective). However, we observe that the deterministic set covering problem, which is known to be $\mathcal{NP}$-hard, can be reduced to one of the subproblems required for this algorithm (see Appendix C). Our computational results show that the need to solve a large number of such difficult integer programming subproblems makes this algorithm prohibitive for the PPSC application. In this chapter, we propose an alternative approach and use the submodularity property of PPSC to solve formulation (3.11), which we describe next.

For each scenario $\omega \in \Omega$, let $\sigma_{\omega}(x)$ denote the number of nodes in V_2 covered by the selection x in the live-arc graph $G_{\omega} = (V_1 \cup V_2, E_{\omega})$. It is known that $\sigma_{\omega}(x)$ is submodular

[99, 38]. Given a set of scenarios Ω and target level τ, we formulate PPSC as

$$\min \quad \sum_{i \in V_1} b_i x_i \tag{3.12a}$$

$$\text{s.t.} \quad \sigma_\omega(x) \geq \tau z_\omega \qquad \omega \in \Omega \tag{3.12b}$$

$$\sum_{\omega \in \Omega} p_\omega z_\omega \geq 1 - \epsilon \tag{3.12c}$$

$$x \in \mathbb{B}^n, z \in \mathbb{B}^{|\Omega|}, \tag{3.12d}$$

where $z_\omega = 1$ implies that for a given x, $\sigma_\omega(x) \geq \tau$ is enforced. Constraint (3.12c) ensures that the probability that $\sigma_\omega(x) \geq \tau$ is at least $1-\epsilon$. Constraint (3.12b) involves a submodular function. To reformulate it using linear inequalities, we introduce additional variables θ_ω that represent the number of covered nodes in V_2 under scenario $\omega \in \Omega$. In what follows, we use the notation $\sigma(x)$ for a given $x \in \mathbb{B}^n$ and $\sigma(X)$ for the corresponding support $X \subseteq V_1$ interchangeably, and the usage will be clear from the context. For a given $\omega \in \Omega$, consider the polyhedron $\mathcal{S}_\omega = \{(\theta_\omega, x) \in \mathbb{R} \times \mathbb{B}^n : \theta_\omega \leq \sigma_\omega(S) + \sum_{j \in V_1 \setminus S} \rho_j^\omega(S) x_j, \forall S \subseteq V_1\}$, where $\rho_j^\omega(S) = \sigma_\omega(S \cup \{j\}) - \sigma_\omega(S)$ is the marginal contribution of adding $j \in V_1 \setminus S$ to the set S. Nemhauser and Wolsey [64] show that when $\sigma_\omega(x)$ is nondecreasing and submodular $\max_x \sigma_\omega(x)$ is equivalent to $\max_{\theta_\omega, x}\{\theta_\omega : (\theta_\omega, x) \in \mathcal{S}_\omega\}$.

Note that we may need an exponential number of inequalities to represent the submodular function using linear inequalities. Instead of adding these inequalities a priori, we follow a delayed cut generation approach that combines Benders decomposition with the probability oracle to solve PPSC. The corresponding relaxed RMP is defined as

$$\min \quad \sum_{i \in V_1} b_i x_i \tag{3.13a}$$

$$\text{s.t.} \quad (\theta_\omega, x) \in \bar{\mathcal{C}} \tag{3.13b}$$

$$\theta_\omega \geq \tau z_\omega \qquad \omega \in \Omega \tag{3.13c}$$

$$\sum_{\omega \in \Omega} p_\omega z_\omega \geq 1 - \epsilon \tag{3.13d}$$

$$x \in \mathbb{B}^n, z \in \mathbb{B}^{|\Omega|}, \theta \in \mathbb{R}_+^{|\Omega|}, \tag{3.13e}$$

where $\bar{\mathcal{C}}$ is the set of feasibility cuts associated with the decision variables (θ_ω, x) for $\omega \in \Omega$. In particular, given incumbent solution, $\bar{x}$, of RMP (3.13), and its corresponding support $\bar{X} = \{i \in V_1 : \bar{x}_i = 1\}$, a submodular feasibility cut [64, 66] is

$$\theta_\omega \leq \sigma_\omega(\bar{X}) + \sum_{j \in V_1 \setminus \bar{X}} \rho_j^\omega(\bar{X}) x_j. \tag{3.14}$$

At each iteration of the algorithm, we solve RMP (3.13) to obtain an incumbent solution $(\bar{x}, \bar{\theta}, \bar{z})$, which is used to generate the submodular cuts (3.14), if necessary. In a related study, Wu and Küçükyavuz [105] apply inequality (3.14) to solve the stochastic influence maximization problem, which aims to find a subset of k nodes to reach the maximum expected number of nodes in a general (non-bipartite) network. Wu and Küçükyavuz [105] give conditions under which inequalities (3.14) are facet defining for $\mathcal{S}_\omega$. We extend the work of [105], and propose a new class of valid inequalities for the bipartite case. Before we give our proposed inequality, we provide a useful definition.

Definition 1. *Given a live-arc graph $G_\omega = (V_1 \cup V_2, E_\omega)$, if there exists an arc $(i, j) \in E_\omega$, where $i \in V_1$ and $j \in V_2$, then we say that j is reachable from i. Given a set of nodes $B \subseteq V_1$, if node $j \in V_2$ is reachable from all nodes in B and $|B| \geq 2$, we say that j is a common node of all nodes in B. Given two sets of nodes $B \subseteq V_1$ and $N \subseteq V_1$, where $B \cap N = \emptyset$, we define $\mathcal{U}_\omega(B, N) = \{j \in V_2 : (i, j) \in E_\omega,\ \forall i \in B;\ (i, j) \notin E_\omega,\ \forall i \in N\}$ as the set of nodes reachable from all nodes in B but not reachable from any node in N. For $k \in V_1$, let $\eta_\omega^k = |\mathcal{U}_\omega(\{k\}, V_1 \setminus \{k\})|$.*

Next we give a new class of valid inequalities.

Proposition 6. *Given $D \subseteq V_1$, and sets $C_1^k \subseteq V_1$, and $C_2^k \subseteq V_2$ for $k = 1, \ldots, c$ for some $c \in \mathbb{Z}_+$ such that $|C_1^k| \geq 2$, each pair of distinct nodes $\{i, j\} \in C_1^k$ satisfies $|\mathcal{U}_\omega(\{i, j\}, V_1 \setminus C_1^k) \cap C_2^k| =: n_\omega(C_1^k)$ for some $n_\omega(C_1^k) \in \mathbb{Z}_+$, and $C_2^i \cap C_2^j = \emptyset$ for all $i = 1, \ldots, c$ and $j = 1, \ldots, c$ with $i \neq j$, the inequality*

$$\theta_\omega \leq \sum_{k=1}^{c} n_\omega(C_1^k)\Big(1 - \sum_{j \in C_1^k} x_j\Big) + \sum_{k \in D} \eta_\omega^k (1 - x_k) + \sum_{j \in V_1} \sigma_\omega(\{j\}) x_j \tag{3.15}$$

is valid for $\mathcal{S}_\omega$.

Proof. Consider a feasible point $(\hat{\theta}_\omega, \hat{x}) \in \mathcal{S}_\omega$. Note that we must have $\hat{\theta}_\omega \leq \sigma_\omega(\hat{X})$ at a feasible point, where $\hat{X} = \{i \in V_1 : \hat{x}_i = 1\}$. Let $C'' \subset \{1, \ldots, c\}$, where $\sum_{j \in C_1^k} x_j > 1$ for each $k \in C''$. Let $C' = \{1, \ldots, c\} \setminus C''$, where $\sum_{j \in C_1^k} x_j \leq 1$ for each $k \in C'$. We create an additional dummy node d in V_1, where $(d, v) \notin E_\omega$ for all $v \in V_2$, $\sigma_\omega(\{d\}) = 0$ and $\sigma_\omega(\hat{X}) = \sigma_\omega(\hat{X} \cup \{d\})$. For each $v \in V_2$, we define $r(v) := \min\{i \in \hat{X} : (i, v) \in E_w\}$, if there exists $(i, v) \in E_\omega$ for some $i \in \hat{X}$, we let $r(v) := d$, otherwise.

Here $r(v) \neq d$ denotes the node that belongs to $\hat{X}$ and can reach $v \in V_2$. Let $R := \cup_{v \in V_2}\{r(v)\}$. Recall the condition that $C_2^i \cap C_2^j = \emptyset$ for all $i, j = 1, \ldots, c$ with $i \neq j$. In other words, each $v \in V_2$ belongs to at most one C_2^k for all $k = 1, \ldots, c$. For each $k \in C''$, since $\sum_{j \in C_1^k} x_j > 1$ and each pair of distinct nodes $\{i, j\} \in C_1^k$ satisfies $|\mathcal{U}_\omega(\{i, j\}, V_1 \setminus C_1^k) \cap C_2^k| = n_\omega(C_1^k)$ for some $n_\omega(C_1^k) \in \mathbb{Z}_+$, there exists $v \in C_2^k$ such that $r(v) \in C_1^k$. Thus, for each $k \in C''$, we define $r_k := \min\{r(v) \in R \cap C_1^k : v \in C_2^k\}$, where r_k denotes the node that belongs to $\hat{X} \cap C_1^k$ and can reach some node in C_2^k. From the previous discussion, r_k exists for all $k \in C''$. Because $\hat{\theta}_\omega \leq \sigma_\omega(\hat{X})$ at a feasible point, we have

$$
\begin{aligned}
\hat{\theta}_\omega &\leq \sigma_\omega(\hat{X}) \\
&= \sum_{j \in \hat{X}} \sigma_\omega(\{j\}) - \sum_{v \in V_2} \sum_{j \in \hat{X} \setminus \{r(v)\}} |v \cap \mathcal{U}_\omega(\{j, r(v)\}, \emptyset)| && (3.16) \\
&= \sum_{j \in V_1} \sigma_\omega(\{j\})\hat{x}_j - \sum_{v \in V_2} \sum_{j \in V_1 \setminus \{r(v)\}} |v \cap \mathcal{U}_\omega(\{j, r(v)\}, \emptyset)|\hat{x}_j && (3.17) \\
&\leq \sum_{j \in V_1} \sigma_\omega(\{j\})\hat{x}_j - \sum_{k \in C''} \sum_{v \in C_2^k} \sum_{j \in V_1 \setminus \{r(v)\}} |v \cap \mathcal{U}_\omega(\{j, r(v)\}, \emptyset)|\hat{x}_j && (3.18) \\
&\leq \sum_{j \in V_1} \sigma_\omega(\{j\})\hat{x}_j - \sum_{k \in C''} \sum_{v \in C_2^k} \sum_{j \in C_1^k \cap \{V_1 \setminus \{r(v)\}\}} |v \cap \mathcal{U}_\omega(\{j, r(v)\}, \emptyset)|\hat{x}_j && (3.19) \\
&\leq \sum_{j \in V_1} \sigma_\omega(\{j\})\hat{x}_j - \sum_{k \in C''} \sum_{v \in C_2^k} \sum_{j \in C_1^k \cap \{V_1 \setminus \{r_k\}\}} |v \cap \mathcal{U}_\omega(\{j, r_k\}, \emptyset)|\hat{x}_j && (3.20) \\
&= \sum_{j \in V_1} \sigma_\omega(\{j\})\hat{x}_j - \sum_{k \in C''} \sum_{j \in C_1^k \cap \{V_1 \setminus \{r_k\}\}} |C_2^k \cap \mathcal{U}_\omega(\{j, r_k\}, \emptyset)|\hat{x}_j && (3.21)
\end{aligned}
$$

$$\leq \sum_{j\in V_1} \sigma_\omega(\{j\})\hat{x}_j - \sum_{k\in C''} \sum_{j\in C_1^k \cap \{V_1\setminus\{r_k\}\}} |C_2^k \cap \mathcal{U}_\omega(\{j, r_k\}, V_1 \setminus C_1^k)|\hat{x}_j \tag{3.22}$$

$$= \sum_{j\in V_1} \sigma_\omega(\{j\})\hat{x}_j - \sum_{k\in C''} \sum_{j\in C_1^k \cap \{V_1\setminus\{r_k\}\}} n_\omega(C_1^k)\hat{x}_j \tag{3.23}$$

$$= \sum_{j\in V_1} \sigma_\omega(\{j\})\hat{x}_j - \sum_{k\in C''} \sum_{j\in C_1^k \cap \{V_1\setminus\{r_k\}\}} n_\omega(C_1^k)\hat{x}_j + \sum_{k\in C''} n_\omega(C_1^k)(1-\hat{x}_{r_k}) \tag{3.24}$$

$$= \sum_{j\in V_1} \sigma_\omega(\{j\})\hat{x}_j - \sum_{k\in C''} \sum_{j\in C_1^k \cap V_1} n_\omega(C_1^k)\hat{x}_j + \sum_{k\in C''} n_\omega(C_1^k) \tag{3.25}$$

$$\leq \sum_{j\in V_1} \sigma_\omega(\{j\})\hat{x}_j + \sum_{k\in C''} n_\omega(C_1^k)\Big(1 - \sum_{j\in C_1^k} \hat{x}_j\Big) + \sum_{k\in C'} n_\omega(C_1^k)\Big(1 - \sum_{j\in C_1^k} \hat{x}_j\Big) \tag{3.26}$$

$$\leq \sum_{j\in V_1} \sigma_\omega(\{j\}) + \sum_{k=1}^{c} n_\omega(C_1^k)\Big(1 - \sum_{j\in C_1^k} \hat{x}_j\Big) + \sum_{k\in D} \eta_\omega^k(1-\hat{x}_k). \tag{3.27}$$

Equality (3.16) follows from the definition of $\sigma_\omega(\hat{X})$ for a given $\hat{X}$. Equality (3.17) holds because $\hat{x}_j = 0$ for $j \in V_1 \setminus \hat{X}$ and $\hat{x}_j = 1$ for $j \in \hat{X}$. Inequality (3.18) follows from the assumptions that $\bigcup_{k=1}^{c} C_2^k \subseteq V_2$, and $C_2^i \cap C_2^j = \emptyset$ for all $i, j = 1, \ldots, c$ with $i \neq j$. Inequality (3.19) follows from $C_1^k \cap \{V_1 \setminus \{r(v)\}\} \subseteq V_1 \setminus \{r(v)\}$ for $k = 1, \ldots, c$ and $v \in V_2$. Inequality (3.20) follows from the definition of r_k for each $k \in C''$. If $r_k \neq r(v)$ for some $k \in C''$ and $v \in C_2^k$, then there is no arc (r_k, v) in E_ω and the value of $|v \cap \mathcal{U}_\omega(\{j, r_k\}, \emptyset)|$ is equal to 0 for $j \in C_1^k \cap \{V_1 \setminus \{r_k\}\}$. We obtain equality (3.21) by reorganizing the terms in inequality (3.20). Inequality (3.22) follows from $r_k \in C_1^k$ and $\mathcal{U}_\omega(\{j, r_k\}, V_1 \setminus C_1^k) \subseteq \mathcal{U}_\omega(\{j, r_k\}, \emptyset)$ for all $k \in C''$ and $j \in C_1^k \cap \{V_1 \setminus \{r_k\}\}$. Equality (3.23) follows from the assumption that each pair of distinct nodes $\{i, j\} \in C_1^k$ satisfies $|\mathcal{U}_\omega(\{i, j\}, V_1 \setminus C_1^k) \cap C_2^k| = n_\omega(C_1^k)$ for all $k = 1, \ldots, c$ and $n_\omega(C_1^k) \in \mathbb{Z}_+$. Equality (3.24) follows from $\hat{x}_{r_k} = 1$. We obtain equality (3.25) by reorganizing the terms in inequality (3.24). Inequality (3.26) follows from the assumption that $\sum_{j\in C_1^k} x_j \leq 1$ for each $k \in C'$. Finally, inequality (3.27) follows from $\eta_\omega^k \geq 0, x_k \in \mathbb{B}$, and $k \in D$. This completes the proof.

□

Example 1. *Let* $V_1 = \{1, 2, 3, 4\}$ *and* $V_2 = \{1, 2, 3, 4, 5, 6\}$. *The bipartite graph associated with this example is depicted in Figure 3.1. Consider the parameters for inequality* (3.15)

given in Table 3.1. The corresponding inequality (3.15) *is*

$$\theta_\omega \leq 5 + 0x_1 + 0x_2 + 0x_3 + x_4, \tag{3.28}$$

which is equivalent to a submodular inequality (3.14) *with* $\bar{X} = \{1, 2, 3\}$.

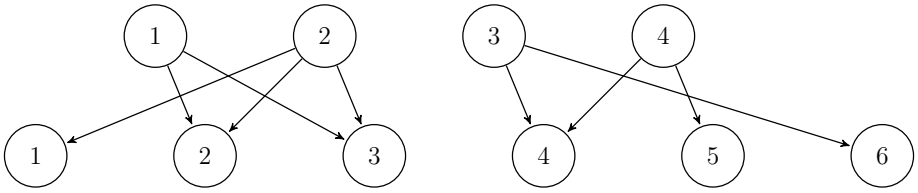

Figure 3.1: An example of a bipartite graph with 4 sets and 6 items.

Table 3.1: The choice of parameters for inequality (3.15) with $D \neq \emptyset$.

$\{C_1^1, C_1^2\}$	$D = \{d_1, d_2\}$
$C_1^1 = \{1, 2\}$	$d_1 = \{2\}$
$n_\omega(C_1^1) = 2$	$\eta_\omega^2 = 1$
$C_2^1 = \{2, 3\}$	
$C_1^2 = \{3, 4\}$	$d_2 = \{3\}$
$n_\omega(C_1^2) = 1$	$\eta_\omega^3 = 1$
$C_2^2 = \{4\}$	

If we let $D = \emptyset$ *for the same choices of* $C_1^k, C_2^k, k = 1, 2$, *then we obtain a facet-defining inequality* $\theta_\omega \leq 3 + x_2 + x_3 + x_4$, *which is stronger than inequality* (3.28). *In addition, this inequality cannot be generated as a submodular inequality* (3.14) *for any selection of* $\bar{X}$. *We formalize this observation next.*

Proposition 7. *Inequalities* (3.15) *subsume the submodular inequalities* (3.14).

Proof. We show that a submodular inequality (3.14) for a given $\bar{X} \subseteq V_1$ can be represented as a corresponding inequality (3.15). To establish this correspondence, let $D = \bar{X}$. From the definition of η_ω^d, for $d \in V_1$, the term $\sum_{d\in D} \eta_\omega^d$ denotes the number of nodes reachable from only one node in $\bar{X}$. Let $\bar{C}$ denote a set of nodes reachable from $\bar{X}$ and at least two nodes in V_1, and let $c = |\bar{C}|$. For $k \in \bar{C}$ let $C_1^k = \{j \in V_1 | (j,k) \in E_\omega\}$, i.e., C_1^k is the set of all nodes that can reach $k \in \bar{C}$, and let $C_2^k = \{k\}$ with $n_\omega(C_1^k) = 1$. The term $\sum_{i=1}^c n_\omega(C_1^i)$ denotes the number of nodes reachable from $\bar{X}$ and at least two nodes in V_1. Next, we show that inequality (3.15) with this choice of D, C_1^k and C_2^k for all $k = 1, \ldots, c$ is equivalent to the submodular inequality.

(i) For each $j \in V_1 \backslash \bar{X}$, the coefficient of x_j in inequality (3.15) is $\sigma_\omega(\{j\}) - \sum_{i=1}^c \sum_{j\in C_1^i} n_\omega(C_1^i)$, where the second term equals the number of nodes reachable from $\bar{X}$ and j. Hence this coefficient is equivalent to the marginal contribution term $\rho_j^\omega(\bar{X})$.

(ii) For each $j \in \bar{X}$, the coefficient of x_j in inequality (3.15) is $\sigma_\omega(\{j\}) - \eta_\omega^j - \sum_{i=1}^c \sum_{j\in C_1^i} n_\omega(C_1^k)$, where the term $\eta_\omega^j + \sum_{i=1}^c \sum_{j\in C_1^i} n_\omega(C_1^i)$ equals the number of nodes reachable from j, i.e., $\sigma_\omega(\{j\})$. Hence, this coefficient is 0.

(iii) The right-hand side of inequality (3.15) is $\sum_{d\in D} \eta_\omega^d + \sum_{i=1}^c n_\omega(C_1^i)$, which equals the number of nodes reachable from $\bar{X}$, i.e., $\sigma_\omega(\bar{X})$.

Hence, any submodular inequality can be represented as an inequality (3.15). Example 1 shows that there are inequalities (3.15) that cannot be written as submodular inequalities (3.14). This completes the proof. □

Next, we provide a necessary condition for inequality (3.15) to be facet defining.

Proposition 8. *Inequality* (3.15) *is facet defining for conv($\mathcal{S}_\omega$) only if* $D = \emptyset$.

Proof. We show that inequality (3.15) with $D = \emptyset$ given by

$$\theta_\omega \leq \sum_{k=1}^{c} n_\omega(C_1^k)\Big(1 - \sum_{j \in C_1^k} x_j\Big) + \sum_{j \in V_1} \sigma_\omega(\{j\}) x_j \tag{3.29}$$

dominates inequality (3.15) with $D \neq \emptyset$. To see this, observe that the coefficients $n_\omega(C_1^k), k = 1, \ldots, c$ and $\sigma_\omega(\{j\}), j \in V_1$ do not depend on D. Hence, for the same choice of $C_1^k, C_2^k, k = 1, \ldots, c$, inequality (3.15) with $D \neq \emptyset$ has the same terms as inequality (3.15) with $D = \emptyset$, as well as the additional term $\sum_{k \in D} \eta_\omega^k (1 - x_k) \geq 0$, because $x_k \in \mathbb{B}$ and $\eta_\omega^k \geq 0$. Hence, we need to have $D = \emptyset$ for inequality (3.15) to be facet defining for $\text{conv}(\mathcal{S}_\omega)$. □

Note that we allow $D \neq \emptyset$ in the definition of inequality (3.15) to be able to show that inequality (3.15) subsumes submodular inequality (3.14). However, we see from the necessary condition in Proposition 8 that it suffices to consider inequalities (3.15) with $D = \emptyset$. Next we give some sufficient conditions for inequality (3.29) to be facet defining for $\text{conv}(\mathcal{S}_\omega)$.

Proposition 9. *Inequality* (3.29) *is facet defining for conv($\mathcal{S}_\omega$) if the following conditions hold:*

(i) $C_1^i \cap C_1^j = \emptyset$ *for each* $i, j = 1, \ldots, c, i \neq j$*,and*

(ii) for each $k = 1, \ldots, c$*, there exists at least one pair of nodes* $\{i, j\} \in C_1^k$ *such that* $\mathcal{U}_\omega(\{i, j\}, \emptyset) = \mathcal{U}_\omega(\{i, j\}, V_1 \setminus C_1^k) \subseteq C_2^k$ *and* $\mathcal{U}_\omega(\{i, r\}, \emptyset) = \mathcal{U}_\omega(\{j, r\}, \emptyset) = \emptyset$ *for all* $r \in V_1 \setminus C_1^k$.

Proof. Note that for $\omega \in \Omega$, $\dim(\mathcal{S}_\omega) = n + 1$. We enumerate $n + 1$ affinely independent points that are on the face defined by inequality (3.29) under conditions (i) and (ii).

Let a pair of nodes $\{f_1^k, f_2^k\} \in C_1^k$ be selected by condition (ii) for all $k = 1, \ldots, c$, where $f_1^k \neq f_2^k$, $\mathcal{U}_\omega(\{f_1^k, f_2^k\}, \emptyset) = \mathcal{U}_\omega(\{i, j\}, V_1 \setminus C_1^k) \subseteq C_2^k$, and $\mathcal{U}_\omega(\{f_1^k, r\}, \emptyset) = \mathcal{U}_\omega(\{f_2^k, r\}, \emptyset) = \emptyset$ for all $r \in V_1 \setminus C_1^k$. Let $\bar{L} = V_1 \setminus \bigcup_{k=1}^c C_1^k$. Based on condition (i), $\sum_{k=1}^c |C_1^k| + |\bar{L}| = n$, which means that each node $i \in V_1$ can only belong either to $\bar{L}$ or to one set C_1^k for some $k = 1, \ldots, c$. We describe $n + 1$ points on the face defined by inequality (3.29) next.

Consider a point $(\theta_\omega, x)^0 = (\sum_{k=1}^c \sigma_\omega(\{f_1^k\}) + \sum_{k=1}^c \sigma_\omega(\{f_2^k\}) - \sum_{k=1}^c n_\omega(C_1^k), \beta_0)$, where $\beta_0 = \sum_{k=1}^c \mathbf{e}_{f_1^k} + \sum_{k=1}^c \mathbf{e}_{f_2^k}$. Recall that for all $k = 1, \ldots, c$, condition (i) ensures that $\mathcal{U}_\omega(\{f_1^k, f_2^k\}, \emptyset) = \mathcal{U}_\omega(\{f_1^k, f_2^k\}, V_1 \setminus C_1^k) \subseteq C_2^k$, so that $n_\omega(C_1^k) = |\mathcal{U}_\omega(\{f_1^k, f_2^k\}, \emptyset)|$. Let $\bar{S}(x) = \{i \in V_1 : x_i = 1\}$. Since $\mathcal{U}_\omega(\{f_1^k, r\}, \emptyset) = \mathcal{U}_\omega(\{f_2^k, r\}, \emptyset) = \emptyset$ for all $r \in V_1 \setminus C_1^k$, we have $\sigma_\omega(\bar{S}(\beta_0)) = \sum_{k=1}^c (\sigma_\omega(\{f_1^k\}) + \sigma_\omega(\{f_2^k\}) - |\mathcal{U}_\omega(\{f_1^k, f_2^k\}, \emptyset)|) = \sum_{k=1}^c (n_\omega(C_1^k) + \sigma_\omega(\{f_1^k\}) - n_\omega(C_1^k) + \sigma_\omega(\{f_2^k\}) - n_\omega(C_1^k))$, hence (θ_ω, β_0) is on the face defined by inequality (3.29).

For $i \in \bar{L}$, let $\beta_i^{\bar{L}} = \sum_{k=1}^c \mathbf{e}_{f_1^k} + \sum_{k=1}^c \mathbf{e}_{f_2^k} + \mathbf{e}_i$. Consider the point $(\theta_\omega, x)^i = (\sum_{k=1}^c \sigma_\omega(\{f_1^k\}) + \sum_{k=1}^c \sigma_\omega(\{f_2^k\}) - \sum_{k=1}^c n_\omega(C_1^k) + \sigma_\omega(\{i\}), \beta_i^{\bar{L}})$ for each $i \in \bar{L}$. Since $\mathcal{U}_\omega(\{f_1^k, r\}, \emptyset) = \mathcal{U}_\omega(\{f_2^k, r\}, \emptyset) = \emptyset$ for all $r \in V_1 \setminus C_1^k$, we have $\sigma_\omega(\bar{S}(\beta_i^{\bar{L}})) = \sigma_\omega(\bar{S}(\beta_0)) + \sigma_\omega(\{i\}) = \sum_{k=1}^c \sigma_\omega(\{f_1^k\}) + \sum_{k=1}^c \sigma_\omega(\{f_2^k\}) - \sum_{k=1}^c |\mathcal{U}_\omega(\{f_1^k, f_2^k\}, \emptyset)| + \sigma_\omega(\{i\}) = \sum_{k=1}^c (n_\omega(C_1^k) + \sigma_\omega(\{f_1^k\}) - n_\omega(C_1^k) + \sigma_\omega(\{f_2^k\}) - n_\omega(C_1^k)) + \sigma_\omega(\{i\})$, hence $(\theta_\omega^i, \beta_i^{\bar{L}})$ for all $i \in \bar{L}$ are on the face defined by inequality (3.29).

Let $\bar{C} = \{1, \ldots, c\}$. For $i \in C_1^k$, $k = 1, \ldots, c$, let $\beta_i^k = \mathbf{e}_i + \sum_{j \in \bar{C} \setminus \{k\}} \mathbf{e}_{f_1^j} + \sum_{j \in \bar{C} \setminus \{k\}} \mathbf{e}_{f_2^j}$. Consider the point $(\theta_\omega, x)^{ik} = (\sum_{j \in \bar{C} \setminus \{k\}} (\sigma_\omega(\{f_1^k\}) + \sigma_\omega(\{f_2^k\}) - n_\omega(C_1^j)) + n_\omega(C_1^k) + \sigma_\omega(\{i\}) - n_\omega(C_1^k), \beta_i^k)$ for each $i \in C_1^k$ and $k = 1, \ldots, c$. For $i \in C_1^k$, $k = 1, \ldots, c$, condition (ii) ensures that $\mathcal{U}_\omega(\{i, f_1^j\}, \emptyset) = \mathcal{U}_\omega(\{i, f_2^j\}, \emptyset) = \emptyset$ for all $j = 1, \ldots, c$ and $j \neq k$. We have $\sigma_\omega(\bar{S}(\beta_i^k)) = \sigma_\omega(\bar{S}(\beta_0)) - \sigma_\omega(\{f_1^k\}) - \sigma_\omega(\{f_2^k\}) + n_\omega(C_1^k) + \sigma_\omega(\{i\}) - n_\omega(C_1^k) = \sum_{j \in \bar{C} \setminus \{k\}} (n_\omega(C_1^j) + \sigma_\omega(\{f_1^j\}) - n_\omega(C_1^j) + \sigma_\omega(\{f_2^j\}) - n_\omega(C_1^j)) + n_\omega(C_1^k) + \sigma_\omega(\{i\}) - n_\omega(C_1^k)$, hence $(\theta_\omega, \beta_i^k)^{ik}$ for all $i \in C_1^k$ and $k = 1, \ldots, c$ are on the face defined by inequality (3.29).

We represent the points corresponding to $x = \beta_i$ for $i \in \bar{L} \cup \{0\}$ and $x = \beta_i^k$ for $i \in C_1^k$, $k = 1, \ldots, c$ in the rows of the matrix given in Table 3.2. Let $\bar{L} := \{1^0, 2^0, \ldots, |\bar{L}|^0\}$, and $C_1^k = \{1^k, 2^k, \ldots, |C_1^k|\}$ for $k = 1, \ldots, c$. The columns of Table 3.2 are reordered according to the indices of x identified in the column names. From this matrix representation, it is easy to see that these $n + 1$ rows are affinely independent. This completes the proof.

□

Algorithm 6 describes a sampling-based method to solve PPSC by using inequalities (3.14)

Table 3.2: The matrix of $n+1$ affinely independent points

	1^1	...	f_1^1	...	f_2^1	...	$\lvert C_1^1\rvert^1$	...	1^k	...	f_1^k	...	f_2^k	...	$\lvert C_1^k\rvert^k$	...	1^c	...	f_1^c	...	f_2^c	...	$\lvert C_1^c\rvert^c$	1^0	...	i^0	...	$\lvert\bar{L}\rvert^0$
β_0	0	...	1	...	1	...	0	...	0	...	1	...	1	...	0	...	0	...	1	...	1	...	0	0	...	0	...	0
$\beta_1^{\bar{L}}0$	0	...	1	...	1	...	0	...	0	...	1	...	1	...	0	...	0	...	1	...	1	...	0	1	...	0	...	0
⋮	⋮	...	⋮	...	⋮	...	⋮	...	⋮	...	⋮	...	⋮	...	⋮	...	⋮	...	⋮	...	⋮	...	⋮	⋮	...	⋮	...	⋮
$\beta_i^{\bar{L}}0$	0	...	1	...	1	...	0	...	0	...	1	...	1	...	0	...	0	...	1	...	1	...	0	0	...	1	...	0
⋮	⋮	...	⋮	...	⋮	...	⋮	...	⋮	...	⋮	...	⋮	...	⋮	...	⋮	...	⋮	...	⋮	...	⋮	⋮	...	⋮	...	⋮
$\beta_{\lvert\bar{L}\rvert}^{\bar{L}}0$	0	...	1	...	1	...	0	...	0	...	1	...	1	...	0	...	0	...	1	...	1	...	0	0	...	0	...	1
$\beta_1^1 1$	1	...	0	...	0	...	0	...	0	...	1	...	1	...	0	...	0	...	1	...	1	...	0	0	...	0	...	0
⋮	⋮	...	⋮	...	⋮	...	⋮	...	⋮	...	⋮	...	⋮	...	⋮	...	⋮	...	⋮	...	⋮	...	⋮	⋮	...	⋮	...	⋮
$\beta_{f_1^1}^1$	0	...	1	...	0	...	0	...	0	...	1	...	1	...	0	...	0	...	1	...	1	...	0	0	...	0	...	0
⋮	⋮	...	⋮	...	⋮	...	⋮	...	⋮	...	⋮	...	⋮	...	⋮	...	⋮	...	⋮	...	⋮	...	⋮	⋮	...	⋮	...	⋮
$\beta_{f_2^1}^1$	0	...	0	...	1	...	0	...	0	...	1	...	1	...	0	...	0	...	1	...	1	...	0	0	...	0	...	0
⋮	⋮	...	⋮	...	⋮	...	⋮	...	⋮	...	⋮	...	⋮	...	⋮	...	⋮	...	⋮	...	⋮	...	⋮	⋮	...	⋮	...	⋮
$\beta_{\lvert C_1^1\rvert^1}^1$	0	...	0	...	0	...	1	...	0	...	1	...	1	...	0	...	0	...	1	...	1	...	0	0	...	0	...	0
⋮	⋮	...	⋮	...	⋮	...	⋮	...	⋮	...	⋮	...	⋮	...	⋮	...	⋮	...	⋮	...	⋮	...	⋮	⋮	...	⋮	...	⋮
$\beta_{1^k}^k$	0	...	1	...	1	...	0	...	1	...	0	...	0	...	0	...	0	...	1	...	1	...	0	0	...	0	...	0
⋮	⋮	...	⋮	...	⋮	...	⋮	...	⋮	...	⋮	...	⋮	...	⋮	...	⋮	...	⋮	...	⋮	...	⋮	⋮	...	⋮	...	⋮
$\beta_{f_1^k}^k$	0	...	1	...	1	...	0	...	0	...	1	...	0	...	0	...	0	...	1	...	1	...	0	0	...	0	...	0
⋮	⋮	...	⋮	...	⋮	...	⋮	...	⋮	...	⋮	...	⋮	...	⋮	...	⋮	...	⋮	...	⋮	...	⋮	⋮	...	⋮	...	⋮
$\beta_{f_2^k}^k$	0	...	1	...	1	...	0	...	0	...	0	...	1	...	0	...	0	...	1	...	1	...	0	0	...	0	...	0
⋮	⋮	...	⋮	...	⋮	...	⋮	...	⋮	...	⋮	...	⋮	...	⋮	...	⋮	...	⋮	...	⋮	...	⋮	⋮	...	⋮	...	⋮
$\beta_{\lvert C_1^k\rvert^k}^k$	0	...	1	...	1	...	0	...	0	...	0	...	0	...	1	...	0	...	1	...	1	...	0	0	...	0	...	0
⋮	⋮	...	⋮	...	⋮	...	⋮	...	⋮	...	⋮	...	⋮	...	⋮	...	⋮	...	⋮	...	⋮	...	⋮	⋮	...	⋮	...	⋮
$\beta_{1^c}^c$	0	...	1	...	1	...	0	...	0	...	1	...	1	...	0	...	1	...	0	...	0	...	0	0	...	0	...	0
⋮	⋮	...	⋮	...	⋮	...	⋮	...	⋮	...	⋮	...	⋮	...	⋮	...	⋮	...	⋮	...	⋮	...	⋮	⋮	...	⋮	...	⋮
$\beta_{f_1^c}^c$	0	...	1	...	1	...	0	...	0	...	1	...	1	...	0	...	0	...	1	...	0	...	0	0	...	0	...	0
⋮	⋮	...	⋮	...	⋮	...	⋮	...	⋮	...	⋮	...	⋮	...	⋮	...	⋮	...	⋮	...	⋮	...	⋮	⋮	...	⋮	...	⋮
$\beta_{f_2^c}^c$	0	...	1	...	1	...	0	...	0	...	1	...	1	...	0	...	0	...	0	...	1	...	0	0	...	0	...	0
⋮	⋮	...	⋮	...	⋮	...	⋮	...	⋮	...	⋮	...	⋮	...	⋮	...	⋮	...	⋮	...	⋮	...	⋮	⋮	...	⋮	...	⋮
$\beta_{\lvert C_1^c\rvert^c}^c$	0	...	1	...	1	...	0	...	0	...	1	...	1	...	0	...	0	...	0	...	0	...	1	0	...	0	...	0

or (3.15) as feasibility cuts. The proposed algorithm includes the Benders phase (Lines 2-14) and the oracle phase (Lines 15-18). Algorithm 6 starts with a given set of feasibility cuts, $\bar{\mathcal{C}}$. In the Benders phase, master problem (3.13) provides an incumbent solution $(\bar{x}, \bar{\theta}, \bar{z})$ at each iteration (Line 3). For each scenario, the incumbent solution $(\bar{x}, \bar{\theta}, \bar{z})$ is used for checking feasibility (Line 4). If the condition in Line 4 is not satisfied, then $(\bar{x}, \bar{\theta}, \bar{z})$ is infeasible for the sample approximation problem. In this situation, we add a feasibility cut (3.14) or (3.15) for ω under the condition in Line 9. The Benders phase terminates when the condition in Line 4 is satisfied. The optimal solution given by the Benders phase to the sample approximation problem is then checked for feasibility with respect to the true distribution, by calling the subroutine FeasibilityCut$(\bar{x}, \kappa, \bar{\mathcal{C}})$ in the oracle phase (Lines 15-18). We use inequality (3.5) with $\kappa(J_0) \leq \kappa$ as the feasibility cut to cut off infeasible $\bar{x}$ until master problem (3.13) provides a truly feasible solution to the original (non-sampled) problem.

For a given incumbent solution $\bar{x}$ with $\bar{X} = \{i \in V_1 : \bar{x}_i = 1\}$, we generate the corresponding violated submodular inequality (3.14) as in [105], if infeasible. Next we describe how to generate a violated new valid inequality (3.29) with $D = \emptyset$ (due to the necessary facet condition in Proposition 8), in polynomial time for a given infeasible solution. Consider the case that a node in V_2 is a common node for at least two nodes in V_1 and at least one node in $\bar{X}$. We find the set of nodes in V_2 reachable from at least two nodes in V_1 and at least one node in $\bar{X}$ by depth-first search, with the worst case complexity $\mathcal{O}(nm)$. Then, let V_2' be a subset of nodes in V_2, where each $j \in V_2'$ is reachable from at least two nodes in V_1 and at least one node in $\bar{X}$. For $k \in V_2'$ let $V_k \subseteq V_1$ denote a set of nodes that $k \in \mathcal{U}_\omega(V_k, V_1 \setminus V_k)$. Note that V_k can be obtained by solving a reachability problem to find which nodes in V_1 can reach node $k \in V_2'$. For each $k \in V_2'$, we let $C_1^k = V_k$ and $C_2^k = \{k\}$ with $n_\omega(C_1^k) = 1$. The complexity of generating C_1^k for all $k = 1, \ldots, |V_2'|$ is $\mathcal{O}(m|V_2'|)$. Thus, a violated inequality (3.29) can be generated in polynomial time.

Finally, for PPSC with the linear threshold model, given a set of sampled scenarios, we observe that a polynomial number of submodular inequalities (3.14) or inequalities (3.15) is sufficient to reach an optimal solution of the master problem (3.13). In the following

Algorithm 6: Sampling-Based Delayed Constraint Generation Algorithm with a Probability Oracle for PPSC

Input: $\kappa \in \{1, 2\}$. Start with $\bar{\mathcal{C}} = \{0 \leq \theta_\omega \leq m, \omega \in \Omega\}$;
while $True$ **do**
    Solve master problem (3.13) and obtain an incumbent solution $(\bar{x}, \bar{\theta}, \bar{z})$;
    if $\sum_{\omega \in \Omega}\{p_\omega : \sigma_\omega(\bar{x}) \geq \tau\} \geq 1 - \epsilon$ **then**
        break;
    end
    else
        for $\omega \in \Omega$ **do**
            if $\tau > \sigma_\omega(\bar{x})$ *and* $\theta_\omega > \sigma_\omega(\bar{x})$ **then**
                Add a feasibility cut (3.14) or (3.15) to $\bar{\mathcal{C}}$ in master problem (3.13);
            end
        end
    end
end
while $\mathcal{A}(\bar{x}) < 1 - \epsilon$ **do**
    Call FeasibilityCut$(\bar{x}, \kappa, \bar{\mathcal{C}})$;
    Solve master problem (3.13) and obtain an incumbent solution $\bar{x}$;
end
Output $\bar{x}$ as an optimal solution.

propositions, we summarize this result and its consequence of providing a compact MIP to solve PPSC with the linear threshold model.

Proposition 10. *For PPSC with the linear threshold model, adding the submodular inequalities* (3.14) *with* $\bar{X} = \emptyset$*, which are equivalent to* (3.29) *for any choice of parameters, to the set* $\bar{\mathcal{C}}$ *for all* $\omega \in \Omega$ *is sufficient to reach an optimal solution of the master problem* (3.13)*.*

Proof. In the live-arc graph scenario generation method proposed by Kempe et al. [38] for the linear threshold model, each node $j \in V_2$ has at most one incoming arc from a node $i \in V_1$ for each scenario $\omega \in \Omega$. Therefore, if $j \in V_2$ is reachable from $i \in V_1$, then j is not reachable from any $i' \in V_1 \setminus \{i\}$. Therefore, for any choice of $c, C_1^k, C_2^k, k = 1, \ldots, c$ for inequality (3.29), we must have $n_\omega(C_1^k) = 0$ for $k = 1, \ldots, c$. In other words, there can be no common nodes in V_2 that are reachable from any two distinct nodes in V_1. Therefore, the submodular inequalities (3.14) with $\bar{X} = \emptyset$ are equivalent to the new inequalities (3.29) for any choice of parameters, and they are given by

$$\theta_\omega \leq \sum_{i \in V_1} \sigma_\omega(\{i\}) x_i. \tag{3.30}$$

Any submodular inequality (3.14) with $\bar{X} \neq \emptyset$ given by

$$\theta_\omega \leq \sigma_\omega(\bar{X}) + \sum_{j \in V_1 \setminus \bar{X}} \rho_j^\omega(\bar{X}) x_j = \sum_{i \in \bar{X}} \sigma_\omega(\{i\}) + \sum_{j \in V_1 \setminus \bar{X}} \sigma_\omega(\{j\}) x_j,$$

is dominated by inequality (3.30). This completes the proof. □

Proposition 11. *For PPSC under the linear threshold model, given a set of scenarios* Ω*, the master problem formulation* (3.13)*, and the deterministic equivalent formulation* (3.11) *can be reduced to the following formulation in* (x, z)*-space*

$$\min \quad \sum_{i \in V_1} b_i x_i \tag{3.31a}$$

$$s.t. \quad \sum_{i \in V_1} \sigma_\omega(\{i\}) x_i \geq \tau z_\omega \qquad \omega \in \Omega \tag{3.31b}$$

$$\sum_{\omega \in \Omega} p_\omega z_\omega \geq 1 - \epsilon \tag{3.31c}$$

$$x \in \mathbb{B}^n, z \in \mathbb{B}^{|\Omega|}. \tag{3.31d}$$

Proof. In Proposition 10, we show that adding inequalities (3.30) to the master problem (3.13) as feasibility cuts is sufficient to capture the submodular coverage function $\sigma_\omega(x)$. Then, the θ_ω variables can be projected out from the formulation using inequalities (3.30) and (3.13c), leading to inequalities (3.31b) and the formulation (3.31).

Next we show that the deterministic equivalent formulation (3.11) can be reduced to formulation (3.31) in (x, z)-space for the linear threshold model. From the definition of t_{ij}^ω for all $i \in V_1$, $j \in V_2$ and $\omega \in \Omega$, where $t_{ij}^\omega = 1$ if arc $(i, j) \in E_\omega$ for $\omega \in \Omega$, and $t_{ij}^\omega = 0$ otherwise, we have $\sigma_\omega(\{i\}) = \sum_{j \in V_2} t_{ij}^\omega$ for all $i \in V_1$ and $\omega \in \Omega$, because there is only one incoming arc to node $j \in V_2$ in every scenario $\omega \in \Omega$ in a linear threshold model [38]. In formulation (3.11), summing the constraints (3.11b) over all $i \in V_2$, we obtain $\sum_{i \in V_2} \sum_{j \in V_1} t_{ij}^\omega x_j \geq \sum_{i \in V_2} y_i^\omega, \forall \omega \in \Omega$, which is equivalent to

$$\sum_{j \in V_1} \sigma_\omega(\{j\}) x_j \geq \sum_{i \in V_2} y_i^\omega, \forall \omega \in \Omega. \tag{3.32}$$

Now we can project out the y variables using the constraints (3.32) and (3.11c), and obtain the constraints (3.31b) and the formulation (3.31). This completes the proof. □

3.4 Computational Experiments

In this section, we report our experiments with PPSC to demonstrate the effectiveness of our proposed methods. All methods are implemented in C++ with IBM ILOG CPLEX 12.7 Optimizer. All experiments were executed on a Windows 8.1 operating system with an Intel Core i5-4200U 1.60 GHz CPU, 8 GB DRAM, and x64 based processor. For the master problem of the decomposition algorithms and the deterministic mixed integer programming models, we specify the MIP search method as traditional branch-and-cut with the lazycallback function of CPLEX. We set the number of threads to one. CPLEX presolve process is turned off for the traditional branch-and-cut for solving the decomposition algorithms. The

relative MIP gap tolerance of CPLEX is set to the default value, so a feasible solution which has an optimality gap of $10^{-4}\%$ is considered optimal. The time limit is set to one hour.

Our dataset is motivated by human sexual contact network (human interaction network) introduced in [28, 68]. This class of social networks is represented as a bipartite graph, where V_1 and V_2 denote the groups of different genders and arcs denote the connections between males and females. Note that, in this context, it is natural to assume that $|V_1|$ is approximately equal to $|V_2|$.

We generate a complete bipartite graph with arcs from all nodes $i \in V_1$ to all nodes $j \in V_2$. We partition the nodes in V_1 into two sets V_1^1 and V_1^2, where each node $i \in V_1^1$ can cover a higher expected number of items than each node $j \in V_1^2$. Our computational experiments include two parts. In the first part of our computational study, we test the exact delayed constraint generation algorithm given in Algorithm 4, and the sampling-based approach described in Algorithm 6 to solve PPSC under the independent probability coverage model. In the second part, we compare the exact delayed constraint generation algorithm and the deterministic equivalent MIP formulation (3.10) to solve PPSC under the linear threshold model. In addition, compact MIP model (3.31) described in Proposition 11 is applied to PPSC under the linear threshold model.

3.4.1 PPSC under the Independent Probability Coverage Model

In this subsection, we report our experiments with the independent probability coverage model. Recall that for the independent probability coverage model, $P(x,i) = 1 - \prod_{j \in V_1}(1 - a_{j,i}x_j)$ is used for calculating $\mathcal{A}(x)$, where $a_{u,i}$ denotes an independent probability that the set u can cover the item i with probability $a_{u,i}$. Because the corresponding model (3.9) is highly nonlinear, we do not attempt to solve it for the independent probability coverage model. We generate a complete bipartite graph where each arc (i,j) is assigned an independent probability a_{ij} of being live for all nodes $i \in V_1, j \in V_2$. We consider the case that the expected number of covered items for each $i \in V_1^1$ is $20\% \pm 2\%$ of the total number of nodes in V_2, and the expected number of covered items for each $i \in V_1^1$ is $2\% \pm 2\%$ of the total

number of nodes in V_2. In particular, we let $a_{ij} = 0.18 + i \times (0.22 - 0.18)/|V_1^1|$ for each $i \in V_1^1$, and $a_{ij} = (i - |V_1^1|) \times (0.04)/|V_1^2|$ for each $i \in V_1^2$, where we let $V_1^2 = \{|V_1^1| + 1, \ldots, n\}$. The size of bipartite graphs is $|V| \in \{60, 90, 120\}$. Unless otherwise noted, we let $n = m = |V|/2$ and $n = |V_1^1| + |V_1^2|$. We let $|V_1^1| = 10$ for all instances, and $|V_1^2| = n - 10$. We set the target $\tau = 0.6m$. The risk level is set as $\epsilon \in \{0.0125, 0.025, 0.05\}$. The objective function coefficients are set as $b_i \in [1, \bar{b}]$ for each $i \in V_1$, where $\bar{b} = \{1, 100\}$. Note that for $\bar{b} \neq 1$, we set $b_i = i\bar{b}/|V_1^1|$ for $i \in V_1^1$. Since each node $j \in V_1^2$ covers a fewer expected number of items than each node $i \in V_1^1$, we set a lower cost range for $j \in V_1^2$ compared to $i \in V_1^1$, where $b_j \in [1, \bar{b}/2]$ and $b_j = (|V_1| - j - 1)\bar{b}/(2|V_1^2|)$ for $j \in V_1^2$.

We first solve PPSC under the independent probability coverage model exactly by using Algorithm 4, which is referred to as "Oracle". To show the effect of the choice of $\kappa(J_0)$ in inequality (3.5) on the convergence of the algorithm, we study two cases of Oracle depending on the choice of the input parameter κ, i.e., Oracle ($\kappa = 1$) and Oracle ($\kappa = 2$). Table 3.3 provides the comparison between the two methods, column "Cuts" denotes the total number of user cuts added to the master problem and column "Time" denotes the solution time in seconds.

Table 3.3 shows that using the stronger no-good cuts (i.e., Oracle ($\kappa = 2$)) drastically reduces the solution time and the number of cuts required when compared to the traditional no-good cuts (i.e., Oracle ($\kappa = 1$)). None of the instances can be solved within the time limit if the traditional no-good cuts are used, whereas all instances are solved in less than six minutes with the coefficient strengthening for instances with $|V| = 60$, and within 20 minutes for unit-cost instances with $|V| = 90$. Hence it is worthwhile to expend additional computational effort to strengthen inequality (3.5) by using a larger right-hand side ($\kappa(J_0) = 2$ versus $\kappa(J_0) = 1$). We observe that the instances with non-unit costs (i.e., $\bar{b} = 100$) are harder to solve than instances with unit cost (i.e., $\bar{b} = 1$). For $|V| \geq 90$, none of the non-unit cost instances can be solved within the time limit. To test the limitation of Oracle ($\kappa = 2$), we also tested instances with a larger size $|V| > 90$ with the same parameters ϵ and $\bar{b}$ as in Table 3.3. The solution time grows exponentially as $|V|$ increases for Oracle ($\kappa = 2$).

For example, Oracle ($\kappa = 2$) can solve only one instance with $(\bar{b}, \epsilon) = (1, 0.05)$ within the time limit for $|V| = 120$ with 3283 seconds. Based on our experience, the oracle-based exact method can solve instances with unit cost of up to 100 nodes within an hour.

To solve the problem for networks with larger sizes (i.e., $|V| > 100$), we consider the sampling-based approach that exploits the submodular substructure of PPSC. We demonstrate the usage of oracle for checking and fixing the feasibility of the solution given by the sampling-based approach. First, we run the sampling-based methods on the instances with $|V| = 60$ used in Table 3.3 so that we can compare the feasible solution obtained at the end of the sampling-based method with the truly optimal solution obtained by the exact method. In our preliminary computational study, consistent with our observations with the exact method, we see that the instances with $\bar{b} = 100$ are harder to solve than the instances with $\bar{b} = 1$. Hence, for the instances with $\bar{b} = 1$, we generate $|\Omega| = \{100, 500, 1000\}$ equiprobable scenarios, and for $\bar{b} = 100$, we generate fewer equiprobable scenarios ($|\Omega| = \{100, 250, 500\}$). For each combination of $(|V|, \bar{b}, \epsilon, |\Omega|)$, we create three replications of the scenario set and report the average statistics. We consider the sampling-based delayed constraint generation method (Algorithm 6), which is referred to as "DCG" in this subsection. Recall that Algorithm 6 is executed in two phases, the Benders phase, and the oracle phase. In the Benders phase, we apply two types of feasibility cuts, submodular inequality (3.14) (referred to as DCG-Sub) and new valid inequality (3.29) (referred to as DCG-NV), to RMP (3.13). In the oracle phase, we check whether the optimal solution to the sample approximation problem, $\bar{x}$, obtained at the end of the Benders phase of DCG, is feasible for the original problem, by using the polynomial-time DP described in Section 3.3.1. We use Algorithm 5 with $\kappa = 2$ in these experiments to add feasibility cuts (3.5) to RMP (3.13). We also consider the deterministic equivalent problem (3.11) using the linear representation of the chance constraint (referred to as DEP (3.11)). In the case of DEP (3.11), once the sample approximation problem is solved to obtain an optimal solution $\bar{x}$ to the sample approximation problem, we also enter an oracle phase, where we check feasibility by using the polynomial-time DP described in Section 3.3.1 as an oracle. If the current solution is not feasible with respect

Table 3.3: Oracle ($\kappa = 1$) vs. Oracle ($\kappa = 2$) for PPSC with the independent probability coverage model.

			Oracle ($\kappa = 1$)		Oracle ($\kappa = 2$)	
$\|V\|$	$\bar{b}$	ϵ	Time	Cuts	Time	Cuts
60	1	0.0125	≥ 3600	39844	90	4357
		0.025	≥ 3600	44387	12	1404
		0.05	≥ 3600	51004	15	1295
	100	0.0125	≥ 3600	49158	292	6903
		0.025	≥ 3600	46392	320	8075
		0.05	≥ 3600	49895	257	7161
90	1	0.0125	≥ 3600	45913	49	2758
		0.025	≥ 3600	45627	387	8072
		0.05	≥ 3600	46594	1209	12894
	100	0.0125	≥ 3600	42648	≥ 3600	25178
		0.025	≥ 3600	42351	≥ 3600	25253
		0.05	≥ 3600	40234	≥ 3600	24900

to the true distribution, then we add inequality (3.5) with $\kappa(J_0) \leq 2$ to the corresponding deterministic equivalent formulation and re-solve. We repeat this process until a feasible solution is obtained. The feasible solution obtained at the end of the oracle phase may not be optimal with respect to the true distribution. For the instances for which the truly optimal solution is available (from Table 3.3), we provide information on the optimality gap of the feasible solution.

In Table 3.4, we compare the performances of DCG-NV, DCG-Sub, and DEP (3.11) and demonstrate the utility of the oracle in the sampling-based delayed constraint generation algorithm for instances for which we are able to find the optimal solution to the true problem with the exact method. Column "Master" reports the statistics pertaining to the Benders phase of DCG, and column "DEP" denotes the deterministic equivalent problem (3.11). Column "Oracle" reports the statistics pertaining to the oracle phase of DCG and DEP (3.11). Note that we set a one hour time limit for both Master and DEP, and for the oracle phase. In "Master," "DEP" and "Oracle" columns, "Time(u)" denotes the solution time in seconds and notation "(u)" denotes the number of unsolved instances out of the three instances tested for the corresponding setting. In "Master," column "Cuts" denotes number of inequalities (3.14) and (3.29) added to RMP (3.13) for DCG-Sub and DCG-NV, respectively. In "Oracle," column "Cuts" denotes number of inequalities (3.5) added to RMP (3.13) in the oracle phase. Column "Nodes" denotes the number of branch-and-bound nodes traced in the Benders phase. For the instances that do not solve within the time limit, column "Gap" reports the end gap given by $(ub - lb)/ub$, where ub is the objective function value of the best feasible integer solution obtained within the time limit and lb is the best lower bound available within the time limit for the sample approximation problem. We use the oracle phase to check and fix the infeasibility of the solution provided by the Benders phase. Column "Inf" denotes the number of instances, among the instances for which an optimal solution was found in the Benders phase, that provides a solution detected to be infeasible by the oracle. Note that we do not enter the oracle phase unless an optimal solution is found by the Benders phase. Hence, if none of the instances are solved to optimality in the Benders

Table 3.4: Networks with $|V| = 60$ for PPSC with the independent probability coverage model-Sampling.

			DCG-NV									DCG-Sub							DEP (3.11)					
			Master				Oracle					Master				Oracle			DEP			Oracle		
$\bar{b}$	ϵ	$\|\Omega\|$	Time(u)	Cuts	Nodes	Gap(%)	Inf	Time(u)	Cuts	nOpto	Gap(%)	Time(u)	Cuts	Nodes	Gap(%)	Inf	Time(u)	Cuts	Time(u)	Nodes	Gap(%)	Inf	Time(u)	Cuts
		100	≤ 1	369	506	0	3	4	148	0	0	2	601	871	0	3	5	121	21	1304	0	3	34	102
	0.0125	500	15	1010	4648	0	3	50	110	0	0	36	2034	8672	0	3	30	81	462	3070	0	3	2441(2)	9
		1000	473	2408	74345	0	1	110	119	0	0	909	4650	131859	0	1	95	56	2794(2)	8244	27.78	1	(1)	3
		100	2	404	1079	0	2	2	16	0	0	3	600	1874	0	2	3	30	16	1006	0	2	63	27
1	0.025	500	110	1410	31559	0	2	27	1	0	0	201	2314	45543	0	1	31	1	1386	5780	0	1	2489	9
		1000	1678	2565	218824	0	1	290	1	0	0	1563	4612	231118	0	3	124	3	2012(1)	3687	30.83	0	-	-
		100	5	451	3301	0	3	24	135	0	0	9	638	4954	0	3	25	142	26	1123	0	3	233	34
	0.05	500	1985	2420	689696	0	0	-	-	0	0	1301(1)	4189	510209	23.2	0	-	-	1507	8646	0	0	-	-
		1000	1633(2)	4553	567985	26.3	0	-	-	0	0	(3)	7586	323825	36.35	-	-	-	2050(1)	3996	34.21	0	-	-
		100	≤ 1	318	348	0	3	3	119	1	1.01	≤ 1	496	710	0	3	3	150	12	581	0	3	31	41
	0.0125	250	5	550	1795	0	3	6	56	0	0	9	851	3447	0	3	12	88	68	1831	0	3	128	23
		500	37	954	10406	0	3	24	47	0	0	78	1495	19342	0	3	16	74	499	4288	0	3	1636	14
		100	≤ 1	328	802	0	3	2	5	2	5.74	2	481	1537	0	2	2	29	20	1010	0	2	40	9
100	0.025	250	16	620	6439	0	3	11	3	2	3.45	24	1009	10158	0	3	7	5	252	5271	0	3	427	4
		500	190	1077	46666	0	2	19	5	2	3.45	291	1778	79243	0	2	19	6	928	6354	0	2	2895	1
		100	6	347	3828	0	3	10	63	1	3.57	8	584	6376	0	3	11	56	34	1432	0	3	175	24
	0.05	250	147	701	59203	0	2	31	38	1	1.19	130	1171	61976	0	2	21	38	499	9027	0	2	1173	13
		500	1463	1572	468709	0	2	43	7	1	1.19	2757	2464	756766	0	2	95	11	1992(2)	11133	18.04	1	(1)	1

phase, we put a dash (-) under the relevant statistics of the oracle phase. In addition, if all solutions found at the end of the Benders phased are deemed feasible by the oracle phase (indicated by Inf = 0), then we put a dash (-) under the "Time(u)" and "Cuts" columns, because the time to confirm that the given solution is feasible is negligible and no oracle cuts are added in this case. If the oracle phase cannot be completed within the one-hour time limit due to the multiple MIPs that need to be solved after detecting infeasibility and adding no-good cuts, then we report the number of unsolved instances out of the number of instances tested in the oracle phase (given by "Inf") in parentheses "(u)." Recall that for the instances with $|V| = 60$, we are able to obtain the truly optimal solution from the exact method. Thus, for these instances, we are able to calculate the actual gap between the objective function value of the feasible solution given by the sampling-based approach and the truly optimal value given by the exact method. Column "nOpt" denotes the number of instances out of three that do not have the same optimal objective value obtained from the exact method. Column "oGap" denotes the optimality gap between the optimal value given by DCG and the true optimal value given by Oracle ($\kappa = 2$) calculated as $100|(v - v^*)|/v$, where v is the objective function value of the feasible solution obtained from DCG and v^* is the optimal objective function of the truly optimal solution found in Table 3.3. We observe that if an instance is solvable within the time limit, all three methods, DCG-NV, DCG-Sub and DEP (3.11), provide the same objective value at the end of the oracle phase. Therefore, we only show "nOpt" and "oGap" for DCG-NV, which is able to solve most of the instances within the time limit.

First, we compare Tables 3.3 and 3.4. We note that for instances with $|V| = 60$, the exact method using the true distribution can solve most problems faster than the sampling-based method that approximates the true problem with the number of scenarios $|\Omega| \geq 500$. For example, the solution time for the setting with $|V| = 60, \bar{b} = 1, \epsilon = 0.05$ with the exact method, Oracle ($\kappa = 2$), is 15 seconds, but the average solution time with the sampling-based method DCG-NV is 1985 seconds for 500 scenarios and two instances hit the time limit for 1000 scenarios. These results demonstrate that, for smaller networks, an exact method

may be able to solve the problem to true optimality more efficiently than a sampling-based method, which is not able to guarantee optimality.

Next, we compare the performance of DCG-NV, DCG-Sub, and DEP (3.11) for problems with $|V| = 60$. Table 3.4 shows that the solution time increases as ϵ and $|\Omega|$ increase. We also note that the problems with non-unit costs are generally harder to solve. Comparing the solution times, we observe that both versions of DCG (with submodular cuts, or with the new valid inequalities) are generally faster than DEP. In addition, for the instances that DCG can provide an optimal solution within the time limit, DCG-NV is faster than DCG-Sub in most cases. In addition, the columns "Cuts" and "Nodes" show that DCG-NV adds fewer user cuts and traces fewer branch-and-bound nodes than DCG-Sub in most cases. As a result, inequality (3.29), which we prove to be a stronger inequality than inequality (3.14) (Proposition (8)), improves the computational performance of DCG.

Next, we demonstrate the usage of oracle for checking and fixing the feasibility of the solution given by the sampling-based approach. In Oracle, Inf = 0 denotes that there the solution provided at the end of the Benders phase is feasible. The instances that require feasibility cuts (indicated by a positive number in the Inf column) show that although the optimal solution for the sample approximation problem, provided by the Benders phase in DCG-Sub or DCG-NV or by DEP (3.11), is not feasible with respect to the true distribution, the oracle phase fixes the infeasibility in most cases. There are two cases for the DEP (3.11) based method where the oracle phase hits the time limit and hence cannot provide a feasible solution. As expected, a larger number of scenarios better represents the true distribution and in general leads to an increased number of feasible solutions that do not require the oracle phase, although there are exceptions. In general, if an instance has a large number of scenarios and an infeasible solution is detected, then the oracle phase spends more time on finding a feasible solution. There is no obvious trend between risk level ϵ and the number of added oracle cuts.

Regarding the optimality gap due to solving a sample approximation problem instead of the true problem, from nOpt and oGap columns, comparing the optimal solutions provided

by the exact method with those of the sampling-based method for the test instances in Tables 3.3 and 3.4, we observe that the instances with unit cost attain true optimality with zero oGap in Table 3.4 at the end of the oracle phase. In most cases, the solution found at the end of the Benders phase is not feasible and it is corrected during the oracle phase, which leads to feasible solutions. While the sampling-based method only guarantees a feasible solution to the original problem at the termination of the oracle phase, in this set of experiments, all solutions for the unit cost instances turn out to be optimal. We suspect that this 0% gap occurs because the objective function has the same coefficient for all the variables, and so there is a large number of solutions - some feasible, some infeasible - with the same objective function value. For the instances with $\bar{b} = 100$, we observe that oracle plays an important role in fixing the infeasibility in all instances and only two settings $(\bar{b}, \epsilon, |\Omega|) = (100, 0.0125, 250)$ and $(100, 0.0125, 500)$ have zero oGap. In most cases, the feasible solutions for non-unit cost instances are suboptimal with an optimality gap of up to 5.74%.

In Table 3.5, we report our experiments with $|V| = 120$ for which we are not able to obtain a truly optimal solution using the exact method we proposed. Comparing Tables 3.4 and 3.5, we see that the problems are harder for the sampling-based method when the number of nodes increases. The solution time and the number of branch-and-bound nodes increase drastically as $|V|$ increases for most instances. For example, the setting $(|V|, \bar{b}, \epsilon, |\Omega|) = (60, 100, 0.025, 500)$ takes 190 seconds and 46666 branch and bound nodes to solve on average, whereas the setting $(|V|, \bar{b}, \epsilon, |\Omega|) = (120, 100, 0.025, 500)$ takes 1581 seconds and 277612 branch-and-bound nodes on average. Table 3.5 shows the same trend as Table 3.4, where the Benders phase of DCG outperforms DEP with respect to solution time. For these instances, DEP (3.11) cannot solve half of the instances within the time limit, whereas both DCG-NV and DCG-Sub solve most instances to optimality. In addition, DCG-NV runs faster than DCG-Sub. We also demonstrate the usage of oracle for checking and fixing the feasibility of the solution given by Master and DEP. The results show that in almost all settings, except for three, an infeasible solution is provided by the sampling-based approach. Therefore, it is important to use the oracle phase to fix the infeasibility.

The number of infeasible solutions decreases as the number of scenarios increases, however, increasing sample size to reduce the infeasibility issues slows down the solution time of both the Benders and the oracle phases of DCG. As a result, there is a tradeoff between the solution time and the solution accuracy.

In Table 3.6, we investigate the quality of the solution obtained by our proposed method for the instances with $|V| = 120$ that are only solvable by the sampling-based approach. Because we do not have the true optimal solution, we cannot provide exact deterministic optimality gaps. However, we use the approximate method proposed in [55] to estimate the optimality gaps with statistical guarantees. In particular, we use Theorem 4 of [55], with $L = 1, \alpha = \epsilon$. Let $\mathcal{M}$ be the number of replications of the sample approximation problems. In our computational study, for each choice of parameters in Table 3.5, we have $\mathcal{M} = 3$ sample approximation problems. We obtain the optimal objective values of these $\mathcal{M}$ sample approximation problems solved by DCG-NV and report the minimum and maximum among these replications under the "Master" column. Under the column "Oracle," we report the minimum and maximum objective function value of the feasible solution provided by the oracle phase. Note that if an objective value provided at the end of the Benders phase is feasible, then the Benders and oracle phases share the same objective value for the corresponding instance. In Table 3.6, we only report the instances that have at least two out of three sample approximation problems solvable by DCG-NV. (i.e., settings $(\bar{b}, \epsilon, |\Omega|) = (1, 0.025, 1000)$, $(1, 0.05, 1000)$ and $(100, 0.05, 500)$ are not reported). Column "EGap" denotes the estimated gap that is equivalent to $(\overline{ub} - \overline{lb})/\overline{ub}$, where $\overline{ub}$ is the best upper bound obtained by the oracle phase (i.e., Min value under Oracle), and $\overline{lb}$ is the lower bound obtained from the Min value of Master. Luedtke and Ahmed [55] show that for $|\Omega|\epsilon$ large enough so that a normal approximation to a binomial distribution is appropriate (e.g., $|\Omega|\epsilon \geq 5$), the lower bound obtained by taking the minimum of optimal objective function values of the sample approximation problems among the $\mathcal{M}$ replications provides a valid lower bound with probability approximately $1 - (0.5)^{\mathcal{M}}$. Using our method, we are now also able to give a deterministic upper bound on the optimal objective function value. Therefore, the gap reported under the

Table 3.5: Networks with $|V| = 120$ for PPSC with the independent probability coverage model-Sampling.

			DCG-NV							DCG-Sub							DEP (3.11)					
			Master				Oracle			Master				Oracle			DEP			Oracle		
$\bar{b}$	ϵ	$\|\Omega\|$	Time(u)	Cuts	Nodes	Gap(%)	Inf	Time(u)	Cuts	Time(u)	Cuts	Nodes	Gap(%)	Inf	Time(u)	Cuts	Time(u)	Nodes	Gap(%)	Inf	Time(u)	Cuts
		100	≤ 1	695	604	0	2	7	101	≤ 1	871	885	0	2	7	114	43	1099	0	2	72	8
	0.0125	500	180	4069	29550	0	0	-	-	255	6187	31508	0	0	-	-	(3)	4645	26.93	-	-	-
		1000	859	6083	84757	0	0	-	-	1740	9969	126428	0	0	-	-	(3)	1514	36.83	-	-	-
		100	3	836	1192	0	3	13	114	3	849	1496	0	3	21	134	39	620	0	3	171	17
1	0.025	500	866	7084	100719	0	0	-	-	1206	9151	114554	0	0	-	-	(3)	5270	31.35	-	-	-
		1000	(3)	11827	128201	39.59	-	-	-	(3)	16492	83493	42.58	-	-	-	(3)	1281	37.37	-	-	-
		100	17	954	7141	0	3	18	1	18	1282	8112	0	2	13	6	72	1033	0	3	77	7
	0.05	500	2426	4138	246194	0	2	1674	14	2461(1)	5139	354775	20	1	895	1	2415	2079	0	3	3110(1)	3
		1000	2650(2)	8669	192779	30.61	1	(1)	5	(3)	9952	225870	27.45	-	-	-	(3)	955	36	-	-	-
		100	2	778	843	0	3	11	479	3	1070	1372	0	3	20	608	38	1166	0	3	168	72
	0.0125	250	23	1567	6141	0	2	46	26	47	2490	9305	0	2	106	100	549	3439	0	2	1325	11
		500	162	2348	27594	0	3	203	39	326	3823	42848	0	3	413	43	(3)	5492	12.61	-	-	-
		100	4	872	1944	0	3	18	233	5	1058	2353	0	3	40	503	36	1002	0	3	220	65
100	0.025	250	152	1634	44208	0	2	557	117	199	2367	49445	0	2	600	217	1477(1)	9610	7.12	2	1723(1)	12
		500	1581	2671	277612	0	2	1806(1)	35	2561	4533	315038	0	2	1591	23	(3)	4732	19.87	-	-	-
		100	17	865	8679	0	3	123	393	36	1240	11537	0	3	271	507	197	3775	0	3	661	64
	0.05	250	732	1903	212938	0	2	1440(1)	209	1165	2503	235058	0	2	(2)	148	3412(2)	15690	6.38	2	(2)	5
		500	(3)	3216	440073	17.68	-	-	-	(3)	4457	305875	22.8	-	-	-	(3)	5760	24.01	-	-	-

EGap column is the estimated optimality gap with approximately 87.5% confidence. We put "*" on the EGap that for settings that do not satisfy $|\Omega|\epsilon \geq 5$ in which case the approximate probabilistic guarantee on the estimated gap is not valid.

From Table 3.6, we observe that Min/Max values under the Master and Oracle columns are non-decreasing as the number of scenarios increases. For most of the instances with $(\bar{b}, |\Omega|) = (100, 100)$, both minimum and maximum objective function values obtained at the end of the Benders phase are smaller than those obtained at the end of the oracle phase. For these cases, the Benders phase cannot provide even a feasible solution as can be seen from Table 3.5, with Inf = 3. Note that even if minimum objective function value obtained at the end of the Benders phase is the same as that obtained at the end of the oracle phase, the solution given by the Benders phase may not be feasible. For example, the setting $(\bar{b}, \epsilon, |\Omega|) = (1, 0.05, 100)$ has the same objective function value in both phases. However, the corresponding instance in DCG-NV of Table 3.5 has Inf = 3, which indicates that none of the solutions provided by the Benders phase is feasible. From EGap(%), we observe that as we increase the sample size $|\Omega|$, we obtain a tighter gap between the feasible upper bound provided by the oracle phase and the lower bound of the optimal value provided by the Benders phase. For the instances with unit costs and $|\Omega|\epsilon \geq 5$, we have a zero gap, which means that DCG-NV provides a truly optimal solution with probability at least 0.875. For non-unit cost instances with $|\Omega| \geq 250$, we are able to provide 1.75%, 6.06%, and 6.45% EGaps with confidence approximately 0.875 for the settings $(\bar{b}, \epsilon) = (100, 0.0125)$, (100,0.025) and (100,0.05), respectively. Therefore, there is a trade-off between the scenario size, solution time and solution quality. If we use a small sample size, the solution time of the master problem is shorter. While in most cases, small sample size leads to infeasible solutions, because the resulting MIP is smaller, the oracle phase also takes a shorter time to fix the infeasibility. Overall, less time is spent in finding a feasible solution, however the quality of the solution may not be as good as that of a solution obtained by using a larger number of scenarios.

In the final part of this experimental subsection, we also test the influence of a larger

size of $|V_2|$ compared to $|V_1|$ in Table 3.7. To test instances with $|V_2| > |V_1|$, we follow the same experimental scheme as in Table 3.4 but we increase $|V_2|$ from 30 to 60. We observe that DCG can solve most instances within an hour for both Master and Oracle, and DCG-NV outperforms DCG-Sub. Compared to DCG, DEP (3.11) includes more constraints and additional y_i^ω variables for $i \in V_2$ and $\omega \in \Omega$. Therefore increasing $|V_2|$ increases the solution time of DEP significantly as can be seen by comparing the solution times in Tables 3.4 and 3.7. However, the sizes of the problems solved by DCG depend only on $|V_1|$, therefore the solution times of DCG are not sensitive to $|V_2|$. There is a slight increase in the time to check feasibility using the DP, however, this time is negligible. Furthermore, we are also able to use the exact method, Oracle ($\kappa = 2$), to solve each instance optimally since the search space $|V_1|$ is still equal to 30. Hence, as in Table 3.4, we compare the optimality gap (oGap) between the optimal value given by DCG-NV and the optimal value given by the exact method for the instances in Table 3.7. The results show the same trend that the oracle phase fixes the infeasibility of a solution obtained from a sample approximation, but the resulting feasible solution is not necessarily optimal. For example, for the non-unit cost instances we obtain solutions that have up to 8.54% optimality gap with respect to the truly optimal solution.

In conclusion, for instances with small $|V_1|$, it is computationally tractable to obtain a truly optimal solution using the exact method Oracle ($\kappa = 2$). For larger $|V_1|$, an approximate sampling-based method is more effective than an exact method. Furthermore, solving the sample approximation problem with delayed cut generation (DCG) is more effective than solving the corresponding DEP model. The new valid inequalities enhance the performance of DCG when compared to using submodular inequalities. While methods that rely on only solving a sample approximation problem cannot guarantee a feasible solution, our oracle-based method combined with the statistical lower bounds of [55] provides provably feasible solutions to the true problem with low optimality gaps with high confidence.

Table 3.6: Solution quality of DCG-NV for networks with $|V| = 120$.

			Master		Oracle		
$\bar{b}$	ϵ	$\lvert\Omega\rvert$	Min	Max	Min	Max	EGap(%)
1	0.0125	100	5	6	6	6	16.7*
		500	6	6	6	6	0
		1000	6	6	6	6	0
	0.025	100	5	5	6	6	16.7*
		500	6	6	6	6	0
	0.05	100	5	5	5	5	0
		500	5	5	5	5	0
100	0.0125	100	157	160	171	172	8.19*
		250	163	172	171	172	4.68*
		500	168	175	171	178	1.75
	0.025	100	148	156	165	167	10.3*
		250	155	165	165	165	6.06
		500	155	165	165	165	6.06
	0.05	100	140	146	155	155	9.68
		250	145	155	155	155	6.45

Table 3.7: Networks with $|V_1| = 30$ and $|V_2| = 60$ for PPSC with the independent probability coverage model.

			DCG-NV									DCG-Sub							DEP (3.11)					
			Master				Oracle					Master				Oracle			DEP			Oracle		
$\bar{b}$	ϵ	$\lvert\Omega\rvert$	Time(u)	Cuts	Nodes	Gap(%)	Inf	Time(u)	Cuts	nOpto	Gap(%)	Time(u)	Cuts	Nodes	Gap(%)	Inf	Time(u)	Cuts	Time(u)	Nodes	Gap(%)	Inf	Time(u)	Cuts
		100	≤ 1	459	265	0	3	2	130	0	0	3	1097	892	0	3	4	68	23	502	0	3	80	9
	0.0125	500	75	2430	17033	0	0	-	-	0	0	125	4813	19987	0	0	-	-	2775(2)	5004	34.7	0	-	-
		1000	390	4435	51940	0	0	-	-	0	0	863	8722	73915	0	0	-	-	(3)	2142	31.17	-	-	-
		100	2	492	1056	0	3	11	157	0	0	2	937	1274	0	3	16	134	16	302	0	3	191	18
1	0.025	500	291	4322	59958	0	0	-	-	0	0	584	8832	63429	0	0	-	-	3362(2)	9031	28.94	0	-	-
		1000	2684(1)	8951	294443	29.42	0	-	-	0	0	(3)	17066	122811	35.40	-	-	-	(3)	1784	32.12	-	-	-
		100	18	824	7867	0	3	8	13	0	0	18	1186	10361	0	3	13	1	50	917	0	1	103	19
	0.05	500	618	2822	101165	0	3	439	18	0	0	1064	5266	126920	0	2	512	35	936(1)	1260	29.73	0	-	-
		1000	795(1)	5526	175280	20	0	-	-	0	0	1350(1)	11248	134195	27.5	2	(2)	1	(3)	1226	32.8	-	-	-
		100	≤ 1	386	280	0	3	3	139	1	1.19	2	703	546	0	3	4	139	8	242	0	3	65	57
	0.0125	250	5	800	1700	0	3	14	59	0	0	11	1538	3056	0	3	16	75	151	1538	0	3	545	20
		500	48	1374	9657	0	2	49	43	0	0	85	2644	17354	0	2	54	63	1630	4533	0	2	(2)	6
		100	≤ 1	347	518	0	3	2	65	1	8.54	≤ 1	770	908	0	3	3	42	24	699	0	3	49	11
100	0.025	250	15	873	4835	0	2	39	34	0	0	15	1368	5559	0	2	26	91	360	2333	0	2	756	8
		500	155	1462	36737	0	2	95	27	1	1.33	275	2880	63236	0	2	63	57	1899(1)	5700	7.26	1	(1)	2
		100	3	459	2223	0	2	6	10	2	6.06	4	784	2358	0	2	7	9	48	1076	0	3	84	2
	0.05	250	27	865	12720	0	3	49	13	2	6.06	51	1387	18840	0	3	17	25	735	4538	0	3	1650	3
		500	511	1687	116886	0	2	270	7	1	6.06	441	3167	107600	0	2	167	10	(3)	5490	14.52	-	-	-

3.4.2 PPSC under the Linear Threshold Model

In this subsection, we report our experiments with the linear threshold model. Given a complete bipartite graph, we assign a deterministic weight a_{ij} to each arc (i, j) from all nodes $i \in V_1$ to all $j \in V_2$. We let $a_{ij} = 0.9/|V_1^1| - i/(100|V_1^1|)$ for each $i \in V_1^1$, and $a_{ij} = (\sum_{i=1}^{|V_1^1|} i/100)/|V_1^2|$ for each $i \in V_1^2$, which satisfies the requirement of the linear threshold model that $\sum_{i:(i,j)\in E} a_{ij} \leq 1$. Recall that in this model, each node $j \in V_2$ has a random threshold drawn from a uniform distribution [0,1]. We let $n = m = |V|/2$, $|V_1^1| = 10$ for all instances, and $|V_1^2| = n - 10$. We consider risk levels $\epsilon \in \{0.0125, 0.025, 0.05\}$. We set the target $\tau = 0.6m$. The objective function coefficients are set as $b_i \in [1, \bar{b}]$ for each $i \in V_1$, where $\bar{b} \in \{1, 100\}$. For $\bar{b} \neq 1$, we set $b_i = i \times \bar{b}/|V_1^1|$ for $i \in V_1^1$, and $b_j = (|V_1| - j - 1) \times \bar{b}/(|V_1^2| \times 2)$ for $j \in V_1^2$.

For the linear threshold model, $P(x, i) = \sum_{j\in V_1} a_{j,i}x_j$ is used in the DP oracle, $\mathcal{A}(x)$, where $a_{j,i}$ denotes a fixed weight on the arc (j, i). This representation leads to an exact compact mixed-integer linear programming model (3.10). To solve PPSC under the linear threshold model, we apply three methods. We first solve it exactly by using Algorithm 4, which is referred to as "Oracle". We only study the cases of Oracle with $\kappa = 2$ in this subsection. The second method is the deterministic equivalent problem (DEP (3.10)) that uses the true distribution. We use the default setting of CPLEX with a single thread to solve DEP (3.10). The dynamic programming formulation in DEP (3.10) computes the actual probability of covered nodes for a given selection from V_1 instead of sampling from the true distribution. The third method is the sampling-based approach, where we take $|\Omega| = 1000$. We follow Propositions 10 and 11, and use formulation (3.31) (referred to as DEP-S (3.31)), to solve the sample approximation problem. We summarize a comparison between these three methods in Table 3.8. The proposed smaller formulation (3.31) has the best performance for our test instances. Therefore, we no longer compare the performance of the original DCG-NV (or equivalently DCG-Sub) and the larger DEP (3.11) in (x, y, z)-space for the linear threshold models. In the case of DEP-S (3.31), once the sample approximation

problem is solved to obtain a solution $\bar{x}$, we check its feasibility using the oracle phase. We add feasibility cuts as necessary until a feasible solution is reached. Note, again, that once a feasible solution is obtained at the end of the oracle phase, there is no guarantee that this solution is optimal to the true problem.

We observe that the exact method is only able to solve the instances with $|V| \leq 70$. Hence, we only show the results of the instances with $|V| \in \{60, 70\}$ in Table 3.8. Column "Time" denotes the total time of solving DEP-S (3.31) including the oracle phase. We do not report the solution time for both DEP-S (3.31) and the oracle phase separately since both phases can solve all instances extremely fast. Column "Cuts" of DEP-S (3.31) denotes the number of feasibility cuts (3.5) added to DEP-S (3.31) by the oracle phase. A positive value in Cuts indicates that the optimal solution given by DEP-S (3.31) is infeasible with respect to the original problem as detected by the oracle. In Table 3.8, Oracle ($\kappa = 2$) provides solutions for 2 out of 12 the instances. Compared to Oracle, our proposed compact reformulation (DEP (3.10)) solves 9 of the 12 instances optimally under the time limit. Note that both Oracle and DEP (3.10) solve all instances under the true distribution for this dataset. On the other hand, the sampling-based approach, DEP-S (3.31) combined with the oracle phase, provides an approximate solution efficiently. In column oGap, we compare the gap between the truly optimal objective value given by the exact method and the objective value of the solution provided by DEP-S (3.31) after the oracle phase. We put "-" in oGap if the exact method is not able to give the optimal solution within the time limit. For our test instances for which a truly optimal solution is available, we see that the feasible solution found by the sampling-based approach is often optimal, there is only one setting $(\bar{b}, \epsilon) = (100, 0.0125)$, which has a 2.2% optimality gap. In this case, the sampling-based approach cannot guarantee the optimality even we use a large number of scenarios such as $|\Omega| = 1000$. Furthermore, for the linear threshold instances with a large number of scenarios, most solutions to the sample approximation problem are feasible, there are only two settings ($(\bar{b}, \epsilon) = (1, 0.0125)$ and (100,0.0125)) where oracle cuts were necessary to correct infeasibility of the solution given by the sample approximation.

Table 3.8: The Exact and Sampling Methods for PPSC with the linear threshold model.

$\|V\|$	$\bar{b}$	ϵ	Oracle ($\kappa = 2$)		DEP (3.10)	DEP-S (3.31)				
			Time	Cuts	Time	Time	Inf	Cuts	nOpt	oGap(%)
60	1	0.0125	2733	20224	437	2	3	4	0	0
		0.025	1508	16083	440	2	0	0	0	0
		0.05	≥ 3600	25203	668	7	0	0	0	0
	100	0.0125	≥ 3600	33174	2090	4	2	2	1	2.2
		0.025	≥ 3600	32163	1080	4	0	0	0	0
		0.05	≥ 3600	35864	1478	51	0	0	0	0
70	1	0.0125	≥ 3600	27858	2022	≤ 1	0	0	0	0
		0.025	≥ 3600	26977	3453	2	0	0	0	0
		0.05	≥ 3600	24881	3315	28	0	0	0	0
	100	0.0125	≥ 3600	37440	≥ 3600	2	0	0	-	-
		0.025	≥ 3600	37165	≥ 3600	7	0	0	-	-
		0.05	≥ 3600	39837	≥ 3600	55	0	0	-	-

Next, we report our experience with the linear threshold instances with a larger size $|V| = 120$ in Table 3.9. We record the solution time for DEP-S (3.31) and the oracle phase (referred to as Oracle), separately. We consider the number of scenarios with $|\Omega| \in \{100, 500, 1000\}$ and perform three replications of the sample set. For the oracle phase, as in the previous subsection, Inf records the number of infeasible solution detected by the oracle, and column Cuts records the number of inequalities (3.5) added by the oracle phase. In our computational study, we observe that there are instances that cannot reach a feasible solution during the oracle phase within the time limit. Hence, in Oracle, Time(u) records the solution time for those instances that completed this phase, and "(u)" denotes the number of unsolved instances out of the Inf number of instances. If Inf = 0, we put "-" in both Time(u) and Cuts. Since the exact method cannot provide the truly optimal solution within the time limit for the instances in Table 3.9, we also provide the solution quality analysis in this part. For both DEP-S (3.31) and Oracle, we record the minimum and maximum objective function values for the solution of each phase over the three replications, and the resulting estimated optimality gap. Note that both DEP-S (3.31) and Oracle have the same objective value if an objective value provided by the DEP-S (3.31) is feasible.

From Table 3.9, we see that although both Oracle and DEP (3.10) cannot solve any instance with $|V| = 120$ within the time limit, DEP-S (3.31) can solve all instances well within a minute, on average. For a small number of scenarios $|\Omega| = 100$, the solution time is within a second. However, a small number of scenarios may cause infeasible solutions. We observe that out of the six settings with $|\Omega| = 100$, the oracle phase detects infeasibility in five settings (a total number of 13 instances under these settings is infeasible). In addition, for the setting $(\bar{b}, \epsilon) = (100, 0.05)$, the oracle phase fixes the solutions of instances with a larger number of scenarios $|\Omega| \geq 500$. Considering the number of cuts added and the solution time in the oracle phase, we observe that a small number of scenarios may also lead to a large number of oracle cuts, and few instances cannot even be solved during the oracle phase within the time limit (e.g., $(\bar{b}, \epsilon, |\Omega|) = (100, 0.0125, 100)$ and $(100, 0.025, 100)$). Hence, a small sample size may be useful to solve the DEP faster, but more time may be spent to

Table 3.9: Solution analysis of DEP-S (3.31) for networks with $|V| = 120$.

			DEP-S (3.31)			Oracle					
$\bar{b}$	ϵ	$\lvert\Omega\rvert$	Time	Min	Max	Inf	Time(u)	Cuts	Min	Max	EGap(%)
		100	≤ 1	9	9	0	-	-	9	9	0
	0.0125	500	≤ 1	9	9	0	-	-	9	9	0
		1000	3	9	9	0	-	-	9	9	0
		100	≤ 1	8	9	1	≤ 1	2	9	9	11.1*
1	0.025	500	2	9	9	0	-	-	9	9	0
		1000	17	9	9	0	-	-	9	9	0
		100	≤ 1	8	8	3	≤ 1	7	9	9	11.1
	0.05	500	25	9	9	0	-	-	9	9	0
		1000	163	9	9	0	-	-	9	9	0
		100	≤ 1	391	446	3	4(1)	16611	450	452	13.1*
	0.0125	500	5	450	450	0	-	-	450	450	0
		1000	6	450	450	0	-	-	450	450	0
		100	≤ 1	376	409	3	715(1)	24434	450	450	16.4*
100	0.025	500	16	450	450	0	-	-	450	450	0
		1000	67	450	450	0	-	-	450	450	0
		100	≤ 1	360	369	3	14	1070	388	390	7.22
	0.05	500	7	374	398	3	15	61	388	399	3.61
		1000	39	370	404	3	57	76	388	407	4.64

correct the resulting infeasible solutions.

Finally, we demonstrate the solution quality of the linear threshold instances with $|V| = 120$. In Table 3.9, a large value of EGap denotes that the objective value given by DEP-S (3.31) is far from the feasible objective value provided by the oracle. The observation shows that a small number of scenarios, in general, leads to a large estimated gap of EGap $> 10\%$. In Table 3.5, we consider the condition that $|\Omega|\epsilon \geq 5$, and put "*" on the EGap the settings that do not satisfy $|\Omega|\epsilon \geq 5$. The EGap without "*" is valid with probability approximately 0.875 for the corresponding setting. For the instances with unit costs and $|\Omega|\epsilon \geq 5$, we have a zero gap, which means that DEP-S (3.31) provides a truly optimal solution with probability at least 0.875. Using $|\Omega| \geq 500$, DEP-S (3.31) is also able to provide the truly optimal solution with confidence 0.875 for most instances with $\bar{b} = 100$. In summary, for the linear threshold instances with $|\Omega| \geq 500$, DEP-S (3.31) not only solves the problem efficiently but also provides a high-quality solution with an estimated gap no larger than 5%.

Chapter 4

RISK-AVERSE SUBMODULAR MAXIMIZATION PROBLEMS

In this chapter, we consider a class of risk-averse submodular maximization problems (RASM). Suppose that we have a random objective function, which is nondecreasing submodular and higher values of the random objective function are preferred. We measure the risk associated with the random objective function through the *conditional value-at-risk* (CVaR) where larger values indicate less risky random outcomes. That is, given a risk level $\alpha \in (0, 1]$, we aim to maximize the expected value of the worst $100\alpha\%$ of outcomes. We assume that we have an efficient oracle that computes the CVaR of the random objective function exactly. Using this oracle, we propose an exact general method for solving this problem. Moreover, we show that the problem can be solved approximately by a sampling-based approach. We demonstrate the proposed methods on a variant of stochastic set covering problem.

4.1 Introduction

Let $V = \{1, \ldots, n\}$ be a finite set, and Ω be a probability space. We define an outcome mapping $\sigma : 2^{|V|} \times \Omega \to \mathbb{R}$. The random outcome $\sigma(X) : \Omega \to \mathbb{R}$ is defined by $\sigma(X)(\omega) = \sigma(X, \omega)$ for all $X \subseteq V$ and $\omega \in \Omega$. We assume that $\sigma(X)(\omega)$ is a nondecreasing submodular set function. Given $\omega_i \in \Omega$, the mapping $\sigma_i : 2^V \to \mathbb{R}$ is defined by $\sigma_i(X) = \sigma(X, \omega_i)$. Given a vector of $|V|$ binary decision variables $x \in \mathbb{B}^{|V|}$, we define $x_i = 1$ if the i^{th} element of V is selected, $x_i = 0$ otherwise. In what follows, we use the notation $\sigma(x)$ for a given $x \in \mathbb{B}^{|V|}$ and $\sigma(X)$ for the corresponding support $X \subseteq V$ interchangeably. The risk-neutral stochastic submodular maximization problem is

$$\max_{x \in \mathcal{X}} \mathbb{E}[\sigma(x)], \tag{4.1a}$$

where $\mathcal{X}$ represents the deterministic constraints on the variables x and $\mathbb{E}[\sigma(x)]$ represents the expectation of $\sigma(X)$ with respect to ω. Problem (4.1) arises in a wide range of applications, such as social networks, machine learning, and supply chain management. In this chapter, we consider a risk-averse decision maker. We measure the risk associated with $\sigma(x)$ via conditional value-at-risk (CVaR) where larger values correspond to less risky random outcomes. In this context, risk measures are referred to as *acceptability functionals*. Let $[z]_+ = \max(z, 0)$ be the positive part of a number $z \in \mathbb{R}$. Let η be a variable that measures the *value-at-risk* (VaR) at a given risk level. For a given $\bar{x} \in \mathcal{X}$, the value-at-risk at a risk level $\alpha \in (0, 1]$ is defined as

$$\text{VaR}_\alpha(\sigma(\bar{x})) = \max\Big\{\eta : \mathbb{P}(\sigma(\bar{x}) \geq \eta) \geq 1 - \alpha, \eta \in \mathbb{R}\Big\}. \tag{4.2}$$

Based on Rockafellar and Uryasev [81, 82], for a given $\bar{x} \in \mathcal{X}$, the conditional value-at-risk at a risk level $\alpha \in (0, 1]$ is defined as

$$\text{CVaR}_\alpha(\sigma(\bar{x})) = \max\Big\{\eta - \frac{1}{\alpha}\mathbb{E}([\eta - \sigma(\bar{x})]_+) : \eta \in \mathbb{R}\Big\}, \tag{4.3}$$

and it provides the conditional expected value that is no larger than the value-at-risk at the risk level α. In general, the risk level α is small, such as $\alpha = 0.05$ or 0.01. Given a risk level $\alpha \in (0, 1]$, a class of risk-averse submodular maximization problems (RASM) is defined as

$$\max_{x \in \mathcal{X}} \text{CVaR}_\alpha(\sigma(x)). \tag{4.4}$$

Note that if $\alpha = 1$, problem (4.4) is equivalent to problem (4.1), which is $\mathcal{NP}$-hard. Maehara [58] shows the hardness of finding a polynomial time approximation algorithm to problem (4.4). In contrast to finding an approximation algorithm, our main goal is to provide general methods for solving problem (4.4) optimally.

We assume that there exists an efficient oracle that provides an exact value of $\text{CVaR}_\alpha(\sigma(\bar{x}))$ for a given incumbent solution $\bar{x}$. We model problem (4.4) as a two-stage optimization model, where the first-stage problem includes an approximation variable to evaluate the upper bound of $\text{CVaR}_\alpha(\sigma(x))$ for $x \in \mathcal{X}$. Note that for the second-stage subproblem, the value function

$\text{CVaR}_\alpha(\sigma(x))$ with $\alpha \neq 1$ is not submodular for a given $x \in \mathcal{X}$. Thus, the submodular inequality of Nemhauser and Wolsey [64] can only be applied to $\text{CVaR}_1(\sigma(x))$, but not for $\text{CVaR}_\alpha(\sigma(x))$ with $\alpha \neq 1$. In this case, we propose a new valid inequality to solve the problem. Moreover, we introduce a class of lifted inequalities with a sequence of lifting functions. Since the corresponding lifting functions are hard to solve, we propose a heuristic to solve a relaxation of the lifting functions in polynomial time.

We also consider a sampling-based approach to problem (4.4). Under the assumption of a finite probability space, where $\Omega = \{\omega_1, \ldots, \omega_N\}$ for $N \in \mathbb{N}$, two-stage stochastic programming has been widely used for optimizing the CVaR measures. Ahmed [3] and Noyan [69] study the case that only continuous variables appear in the second-stage problem. In this case, the generic Benders-decomposition approach applies to solve such problems. Schultz and Tiedemann [87] study another case that the second-stage subproblem includes the integer variables, and propose a Lagrangian relaxation method. Miller and Ruszczyński [61] study an extended two-stage stochastic programming model, where the uncertainty remains after having the second-stage decision. In this chapter, we consider a two-stage stochastic programming model, where the first-stage master problem is a binary program, and the second-stage subproblem includes a submodular value function. To solve the model, we extend the work of [105], which shows that if the second-stage value function is submodular, then one can apply the submodular inequalities proposed by Nemhauser and Wolsey [64] to solve the risk-neutral problem with delayed cut generation. With a finite number of samples, we model problem (4.4) as a two-stage stochastic programming model that exploits submodularity, and give a decomposition algorithm to solve the problem optimally.

To demonstrate our proposed methods, we consider a risk-averse set covering problem (RASC). Given a collection of n subsets of m items and an integer k, a variant of set covering problem aims to choose at most k subsets among the collection, where the largest number of items are covered by the chosen subsets. Given a risk level α, RASC aims to choose at most k subsets from the collection so that the expected value of the number of covered items with respect to the worst $100\alpha\%$ of the outcomes is maximized. We observe that under the

independent probability and linear threshold models, RASC admits an efficient CVaR oracle. We demonstrate the proposed methods on RASC.

We give an outline of the chapter as follows. In Section 4.2, we introduce the concept of a CVaR oracle for problem (4.4), and formulate the problem as a two-stage optimization model. We introduce a delayed constraint generation algorithm with various classes of valid inequalities to solve the problem under the assumption that an efficient CVaR oracle exists. In addition, we give a two-stage stochastic programming model for problem (4.4) under the assumption of a finite probability space, and propose another delayed constraint generation algorithm to solve the problem. In Section 4.3, we use RASC to demonstrate the proposed methods.

4.2 Models and Methods for RASM

In this section, we study methods for solving problem (4.4). In the first subsection, we consider the concept of a CVaR oracle and provide an exact method with various valid inequalities for problem (4.4). In the second subsection, we give a two-stage stochastic programming model and a method for problem (4.4) under the assumption of a finite probability space.

4.2.1 RASM with an Exact CVaR Oracle

In this subsection, we assume that for a given incumbent solution $\bar{x} \in \mathcal{X}$, we have an oracle that computes $\mathrm{CVaR}_\alpha(\sigma(\bar{x}))$ exactly in polynomial time. In such a case, we solve problem (4.4) without sampling by using a two-stage optimization model and an exact algorithm with various valid inequalities.

Let ψ be a variable that denotes the value of $\mathrm{CVaR}_\alpha(\sigma(x))$ for a solution $x \in \mathcal{X}$. The master problem at an iteration is formulated as

$$\max \quad \psi \tag{4.5a}$$

$$\text{s.t.} \quad (x, \psi) \in \mathcal{C} \tag{4.5b}$$

$$x \in \mathcal{X}, \psi \in \mathbb{R}_+, \tag{4.5c}$$

where $\mathcal{C}$ is a set of optimality cuts to be defined later.

Based on master problem (4.5), we propose a delayed constraint generation algorithm for solving problem (4.4) (see Algorithm 7). Algorithm 7 starts with a set of optimality cuts $\mathcal{C}$ (could be empty). In the while loop, we solve the master problem and get an incumbent solution (Line 3). Based on the incumbent solution, we add an optimality cut to the master problem (Line 4). In this algorithm, ϵ is a user-defined optimality tolerance. Let UB be the upper bound obtained from the optimal objective value of the master problem at each iteration. Let LB be the lower bound obtained from $\text{CVaR}_\alpha(\sigma(\bar{x}))$ for a given incumbent solution $\bar{x} \in \mathcal{X}$. We terminate the algorithm and return the near-optimal solution when the optimality gap is below ϵ.

Algorithm 7: A Delayed Constraint Generation Algorithm with the CVaR oracle

1 Start with an initial set of optimality cuts in $\mathcal{C}$ (could be empty), UB$= \infty$ and LB$= -\infty$;
2 **while** *UB-LB* $\leq \epsilon$ **do**
3 Solve the master problem (4.5) and obtain $(\bar{\psi}, \bar{x})$. Let UB be the upper bound obtained from the optimal objective value of the master problem;
4 Add an optimality cut to $\mathcal{C}$;
5 LB$\leftarrow \text{CVaR}_\alpha(\sigma(\bar{x}))$;
6 **end**
7 Output $\bar{x}$ as the optimal solution.

Note that Maehara [58] shows that $\text{CVaR}_\alpha(\sigma(x))$ is not submodular in x even if $\sigma(x)$ is submodular. We cannot use the submodular inequality of Nemhauser and Wolsey [64] as a class of optimality cuts in $\mathcal{C}$. To solve the problem, we propose various optimality cuts in this subsection. Throughout, we let $\mathbf{e}_j$ be a unit vector of dimension $|V|$ whose jth component

is 1, and let $\mathbf{1}$ be a $|V|$-dimensional vector with all entries equal to 1. For a given incumbent solution, $\bar{x}$, which is a characteristic vector of the set $\bar{X}$, an optimality cut is defined as

$$\psi \leq \mathrm{CVaR}_\alpha(\sigma(\bar{x})) + \sum_{j \in V \setminus \bar{X}} \big(\mathrm{CVaR}_1(\sigma(\bar{x} + \mathbf{e}_j)) - \mathrm{CVaR}_\alpha(\sigma(\bar{x}))\big) x_j. \tag{4.6}$$

Proposition 12. *For a given $\bar{x} \in \mathbb{B}^n$ and its support $\bar{X} \subseteq V$, inequality* (4.6) *is valid for master problem* (4.5)*.*

Proof. Consider a feasible point $(\hat{\psi}, \hat{x})$. Note that we must have $\hat{\psi} \leq \mathrm{CVaR}_\alpha(\hat{x})$ at a feasible point. Let $\hat{X} = \{i \in V : \hat{x}_i = 1\}$. Recall that the definition of CVaR is $\mathrm{CVaR}_\alpha(\sigma(\bar{x})) = \max\left\{\eta - \frac{1}{\alpha}\mathbb{E}([\eta - \sigma(\bar{x})]_+) : \eta \in \mathbb{R}\right\}$ for $\bar{x} \in \mathcal{X}$. Since $\sigma(x)(\omega)$ is nondecreasing in x for all $\omega \in \Omega$, function $\mathrm{CVaR}_\alpha(\sigma(x))$ is also nondecreasing in x.

(i) For the case that $\hat{X} \subseteq \bar{X}$, since $\mathrm{CVaR}_\alpha(\sigma(x))$ is a monotonically nondecreasing function in x and $\hat{x}_j = 0$ for all $j \in V \setminus \bar{X}$, we have $\hat{\psi} \leq \mathrm{CVaR}_\alpha(\sigma(\hat{x})) \leq \mathrm{CVaR}_\alpha(\sigma(\bar{x}))$, which shows that inequality (4.6) is valid for $\hat{X} \subseteq \bar{X}$ because the coefficients of x_j in inequality (4.6) are nonnegative.

(ii) For the case that $\hat{X} \setminus \bar{X} \neq \emptyset$, we select an arbitrary $j' \in \hat{X} \setminus \bar{X}$, where $\hat{x}_{j'} = 1$, and have

$$\begin{aligned}
\hat{\psi} &\leq \mathrm{CVaR}_\alpha(\sigma(\hat{x})) && \text{(4.7a)}\\
&\leq \mathrm{CVaR}_1(\sigma(\hat{x})) && \text{(4.7b)}\\
&\leq \sum_{j \in \hat{X} \setminus \bar{X}} \mathrm{CVaR}_1(\sigma(\hat{x} + \mathbf{e}_j))\hat{x}_j - \sum_{j \in \hat{X} \setminus \bar{X} \cup \{j'\}} \mathrm{CVaR}_1(\sigma(\hat{x}))\hat{x}_j && \text{(4.7c)}\\
&\leq \sum_{j \in V \setminus \bar{X}} \mathrm{CVaR}_1(\sigma(\hat{x} + \mathbf{e}_j))\hat{x}_j - \sum_{j \in V \setminus \bar{X} \cup \{j'\}} \mathrm{CVaR}_1(\sigma(\hat{x}))\hat{x}_j && \text{(4.7d)}\\
&\leq \sum_{j \in V \setminus \bar{X}} \mathrm{CVaR}_1(\sigma(\hat{x} + \mathbf{e}_j))\hat{x}_j - \sum_{j \in V \setminus \bar{X} \cup \{j'\}} \mathrm{CVaR}_\alpha(\sigma(\hat{x}))\hat{x}_j && \text{(4.7e)}\\
&= \left(\sum_{j \in V \setminus \bar{X}} \mathrm{CVaR}_1(\sigma(\hat{x} + \mathbf{e}_j))\hat{x}_j - \sum_{j \in V \setminus \bar{X}} \mathrm{CVaR}_\alpha(\sigma(\hat{x}))\hat{x}_j\right) + \mathrm{CVaR}_\alpha(\sigma(\hat{x}))\hat{x}_{j'} && \text{(4.7f)}
\end{aligned}$$

$$= \mathrm{CVaR}_\alpha(\sigma(\hat{x})) + \sum_{j \in V \setminus \bar{X}} (\mathrm{CVaR}_1(\sigma(\hat{x} + \mathbf{e}_j)) - \mathrm{CVaR}_\alpha(\sigma(\hat{x})))\hat{x}_j. \tag{4.7g}$$

Inequality (4.7b) follows from the definition of CVaR at a risk level $\alpha \in (0, 1]$. Inequality (4.7c) follows from the nondecreasing submodular property that $\mathrm{CVaR}_1(\sigma(\hat{x})) \leq \mathrm{CVaR}_1(\sigma(\hat{x} + \mathbf{e}_{j'}))\hat{x}'_j$ and $\sum_{j \in \hat{X} \setminus \bar{X} \cup \{j'\}} \big(\mathrm{CVaR}_1(\sigma(\hat{x} + \mathbf{e}_j)) - \mathrm{CVaR}_1(\sigma(\hat{x}))\big)\hat{x}_j \geq 0$. Inequality (4.7d) follows from $\hat{X} \subseteq V$, $\mathrm{CVaR}_1(\sigma(\hat{x})) \leq \mathrm{CVaR}_1(\sigma(\hat{x} + \mathbf{e}_{j'}))\hat{x}'_j$ and $\sum_{j \in V \setminus \bar{X} \cup \{j'\}} \big(\mathrm{CVaR}_1(\sigma(\hat{x} + \mathbf{e}_j)) - \mathrm{CVaR}_1(\sigma(\hat{x}))\big)\hat{x}_j \geq 0$. Inequality (4.7e) follows from the definition of CVaR at a risk level α. To obtain equality (4.7f), we add $\mathrm{CVaR}_\alpha(\sigma(\hat{x}))\hat{x}_{j'} - \mathrm{CVaR}_\alpha(\sigma(\hat{x}))\hat{x}_{j'}$ to inequality (4.7e) and reorganize the terms. Finally, equality (4.7g) follows from $\hat{x}_{j'} = 1$. This completes the proof.

□

Next, we also introduce a class of lifted inequalities. Given an incumbent solution $\bar{x}$, which is a characteristic vector of the set $\bar{X}$, we let $j_1, \ldots, j_r$ be an ordering of $V \setminus \bar{X}$, where $r = |V \setminus \bar{X}|$. We consider a valid inequality

$$\psi \leq \mathrm{CVaR}_\alpha(\sigma(\bar{x})) + \sum_{i=1}^{r} \delta_{j_i}(\bar{x}) x_{j_i}, \tag{4.8}$$

where $\delta_{j_t}(\bar{x})$ for $t = 1, \ldots, r$ is an up-lifting function defined as

$$\delta_{j_t}(\bar{x}) = \max \quad \mathrm{CVaR}_\alpha(\sigma(x)) - \sum_{i=1}^{t-1} \delta_{j_i}(\bar{x}) x_{j_i} \tag{4.9a}$$

$$\text{s.t.} \quad x_{j_t} = 1 \tag{4.9b}$$

$$x_{j_i} = 0, \quad i = t+1, \ldots, r \tag{4.9c}$$

$$x \in \mathcal{X}. \tag{4.9d}$$

The up-lifting problem (4.9) is $\mathcal{NP}$-hard in general since it is related to the submodular maximization problem (4.4). Instead of solving problem (4.9) exactly, we propose a relaxation that provides an upper bound. Given an incumbent $\bar{x}$, we define $\bar{\delta}_{j_t}(\bar{x})$ as the relaxation of $\delta_{j_t}(\bar{x})$ for $t = 1, \ldots, r$, where $\bar{\delta}_{j_t}(\bar{x})$ is defined as

$$\bar{\delta}_{j_t}(\bar{x}) = \max \quad \mathrm{CVaR}_\alpha(\sigma(x))$$

$$\text{s.t.} \quad (4.9b), (4.9c)$$
$$x \in \mathbb{B}^{|V|},$$

where we relax constraints $\mathcal{X}$ and remove the term $-\sum_{i=1}^{t-1} \delta_{j_i}(\bar{x}) x_{j_i}$ from the objective function in problem (4.9). Since $\text{CVaR}_\alpha(\sigma(x))$ is monotonically increasing in x, $x_{j_t} = 1$ and $x_{j_i} = 0$ for $i = t+1, \ldots, r$, the feasible solution with $x^*_{j_i} = 1$ for $i = 1, \ldots, t$ has the largest $\text{CVaR}_\alpha(\sigma(x^*))$ for the objective function of $\bar{\delta}_{j_t}(\bar{x})$. Thus, the optimal solution of $\bar{\delta}_{j_t}(\bar{x})$ is $\bar{x}^{j_t} = \mathbf{1} - \sum_{i=t+1}^{r} \mathbf{e}_{j_i}$, where we can use the polynomial time oracle of $\text{CVaR}_\alpha(\sigma(x))$ to get the optimal value of $\bar{\delta}_{j_t}(\bar{x})$ efficiently.

Proposition 13. *Given an incumbent $\bar{x} \in \mathcal{X}$ and an ordering of $V \backslash \bar{X}$ defined as $\{j_1, \ldots, j_r\}$, inequality*

$$\psi \leq CVaR_\alpha(\sigma(\bar{x})) + \sum_{i=1}^{r} \bar{\delta}_{j_i}(\bar{x}) x_{j_i}, \tag{4.11}$$

where $\bar{\delta}_{j_i}(\bar{x}) = CVaR_\alpha(\sigma(\bar{x}^{j_i}))$, is valid for master problem (4.5).

Proof. Since function (4.10) is a relaxation of function (4.9), we have $\bar{\delta}_{j_i}(\bar{x}) \geq \delta_{j_i}(\bar{x})$ for $i = 1, \ldots, r$. This completes the proof. □

Next, given an incumbent $\bar{x}$, we propose a greedy method that generates an inequality (4.11) (see Algorithm 8). In the for loop, Line 4 determines the entry j_i by choosing the candidate s for which $\bar{\delta}_{j_i=s}(\bar{x})$ attains its smallest value, where the candidate s has not been chosen previously.

4.2.2 Approximation of RASM with a Finite Probability Space

Suppose that we use SAA to approximate the probability distribution. Let $(\Omega, 2^\Omega, \mathbb{P})$ be a finite probability space with a set of $N \in \mathbb{N}$ realizations (scenarios) $\Omega = \{\omega_1, \ldots, \omega_N\}$ and $\mathbb{P}(\omega_i) = p_i$ for $i = 1, \ldots N$. Recall that a random outcome mapping $\sigma : 2^V \times \Omega \to \mathbb{R}$, where $\sigma(x) : \Omega \to \mathbb{R}$ is defined by $\sigma(x)(\omega) = \sigma(x, \omega)$ for all $x \in \mathcal{X}$ and $\omega \in \Omega$. Recall that $\sigma_i(x) : 2^V \to \mathbb{R}$ is known to be a nondecreasing submodular function for each $\omega_i \in \Omega$. In

Algorithm 8: GreedyUp-lifting

Input: Given $\alpha \in (0,1]$, and an incumbent solution $\bar{x}$ with a characteristic vector of the set $\bar{X}$. ;

$S \leftarrow V \setminus \bar{X}$;

for $i = 1$ *to* r **do**

  $j_i \leftarrow \arg\min_{s \in S} \bar{\delta}_{j_i = s}(\bar{x})$;

  $x^* \leftarrow \mathbf{1} - \sum_{i=t+1}^{r} \mathbf{e}_{j_i}$;

  $\bar{\delta}_{j_i}(\bar{x}) \leftarrow \bar{\delta}_{j_i}(x^*)$;

  $S \leftarrow S \setminus \{j_i\}$;

end

Output an inequality (4.11) as the solution.

this subsection, for a given extreme point $\bar{x}$ of $\mathcal{X}$ and $\alpha \in (0,1]$, we consider a CVaR model with a finite number of scenarios [81]:

$$\begin{aligned}
\mathrm{CVaR}_\alpha(\sigma(\bar{x})) &= \max\left\{\eta - \frac{1}{\alpha}\mathbb{E}([\eta - \sigma(\bar{x})]_+) : \eta \in \mathbb{R}\right\} && \text{(4.12a)}\\
&= \max_{q \in [N]}\left\{\sigma_q(\bar{x}) - \frac{1}{\alpha}\sum_{i \in [N]} p_i[\sigma_q(\bar{x}) - \sigma_i(\bar{x})]_+\right\} && \text{(4.12b)}\\
&= \max\left\{\eta - \frac{1}{\alpha}\sum_{i \in [N]} p_i w_i : w_i \geq \eta - \sigma_i(\bar{x}) \quad \forall i \in [N], w \in \mathbb{R}^N_+, \eta \in \mathbb{R}\right\}, && \text{(4.12c)}
\end{aligned}$$

where $[N] = \{1, \ldots, N\}$ represents the set of the first N positive integers and $\sigma_q(\bar{x}) = \mathrm{VaR}_\alpha(\sigma(\bar{x}))$ for at least one $\omega_q \in \Omega$. Next, we consider a two-stage stochastic model for problem (4.4) that is based on formulation (4.12).

Using the definition of CVaR described in formulation (4.12), given $\alpha \in (0,1]$ and a finite set of scenarios Ω, we formulate problem (4.4) as

$$\max \quad \eta - \frac{1}{\alpha}\sum_{i \in [N]} p_i w_i \qquad \text{(4.13a)}$$

$$\text{s.t.} \quad w_i \geq \eta - \sigma_i(x) \qquad \forall i \in [N] \tag{4.13b}$$

$$x \in \mathcal{X}, w \in \mathbb{R}^N_+, \eta \in \mathbb{R}. \tag{4.13c}$$

Since $\sigma_i(x)$ is a submodular function for each $\omega_i \in \Omega$, we apply the submodular inequalities proposed by Nemhauser and Wolsey [64] to formulation (4.13). Let θ_i be a variable that approximates $\sigma_i(x)$ for each $\omega_i \in \Omega$. The generic master problem at an iteration is formulated as

$$\max \quad \eta - \frac{1}{\alpha} \sum_{i \in [N]} p_i w_i \tag{4.14a}$$

$$\text{s.t.} \quad w_i \geq \eta - \theta_i \qquad \forall i \in [N] \tag{4.14b}$$

$$(x, \theta) \in \mathcal{C}' \tag{4.14c}$$

$$x \in \mathcal{X}, w \in \mathbb{R}^N_+, \eta \in \mathbb{R}, \theta \in \mathbb{R}^N_+, \tag{4.14d}$$

where θ is a $|\Omega|$-dimensional vector of variables and $\mathcal{C}'$ is a set of optimality cuts. Note that the submodular function appears in constraint (4.13b). We apply the submodular inequality as a class of optimality cuts in $\mathcal{C}'$. Consider the polyhedra $\mathcal{S}_{\omega_i} = \{(\theta_i, x) \in \mathbb{R} \times \{0,1\}^{|V|} : \theta_i \leq \sigma_i(S) + \sum_{j \in V \setminus S} \rho^i_j(S) x_j, \forall S \subseteq V\}$, for $\omega_i \in \Omega$, where $\rho^i_j(S) = \sigma_i(S \cup \{j\}) - \sigma_i(S)$ is the marginal contribution of adding $j \in V \setminus S$ to the set S. For a given incumbent solution, $\bar{x}$, which is a characteristic vector of the set $\bar{X}$, and scenario $\omega_i \in \Omega$, the submodular inequality

$$\theta_i \leq \sigma_i(\bar{X}) + \sum_{j \in V \setminus \bar{X}} \rho^i_j(\bar{X}) x_j, \tag{4.15}$$

is valid for master problem (4.14) since the submodular function $\sigma_i(x)$ is nondecreasing [64]. Next, we introduce a delayed constraint generation algorithm to solve problem (4.13) (see Algorithm 9).

Algorithm 9 starts with a set of optimality cuts (could be empty). In the while loop, we solve master problem (4.14) and get an incumbent solution at each iteration (Line 4). For each scenario, we add the corresponding submodular inequality (4.15) to $\mathcal{C}'$. The variable x' is used to store the incumbent solution generated at the previous iteration. We terminate

Algorithm 9: A Delayed Constraint Generation Algorithm with a Set of Scenarios

Input: Start with an initial set of optimality cuts in $\mathcal{C}'$ (could be empty) ;
$x' \leftarrow$ NULL;
while $x' \neq \bar{x}$ **do**
    Solve master problem (4.14) and obtain an incumbent solution $(\bar{\theta}, \bar{x})$. ;
    for $\omega_i \in \Omega$ **do**
        Add an optimality cut (4.15) to $\mathcal{C}'$ in master problem (4.14);
    end
    $x' \leftarrow \bar{x}$;
end
Output $\bar{x}$ as the optimal solution.

the while loop in Algorithm 9 when the incumbent solution at the current iteration is the same as the incumbent solution provided at the previous iteration.

4.3 An Application: the Risk-Averse Set Covering Problem (RASC)

In this section, we demonstrate the proposed methods on a risk-averse set covering problem (RASC). Given an integer k, a set of items $V_2 := \{1, \ldots, m\}$ and a collection of n subsets $S_j \subseteq V_1, j \in V_1 := \{1, \ldots, n\}$ such that $\cup_{j=1}^{n} S_j = V_2$, a variant of set covering aims to choose k subsets from V_1 that covers the largest number of items in V_2 [23, 24]. Suppose that there is uncertainty on whether a chosen subset can cover an item [99]. Given a risk level α, RASC aims to choose at most k subsets from the collection so that the expected value of the number of covered items with respect to the worst $100\alpha\%$ of the outcomes is maximized.

We represent RASC on a bipartite graph $G = (V_1 \cup V_2, E)$. There are two groups of nodes V_1 and V_2 in G, where all arcs in E are from V_1 to V_2. Node $i \in V_1$ represents set S_i and nodes in $j \in V_2$ represent the items. There exists an arc $(i, j) \in E$ representing the covering relationship if $j \in S_i$ for $i \in V_1$. In RASC, there is uncertainty on whether an arc appears

in the graph. In this chapter, we consider two types of RASC:

Risk-Averse Set Covering with Independent Probability Coverage: In this model, each node j has an independent probability a_{ij} of being covered by node $i \in V_1$ for $j \in S_i$.

Risk-Averse Set Set Covering with Linear Thresholds: In the linear threshold model of Kempe et al. [38], each arc $(i,j) \in E$ has a deterministic weight $0 \leq a_{ij} \leq 1$, such that for all nodes $j \in V_2$, $\sum_{i:(i,j)\in E} a_{ij} \leq 1$. In addition, each node $j \in V_2$ selects a threshold $\nu_j \in [0,1]$ *uniformly at random.* A node $j \in V_2$ is covered if sum of the weights of its selected neighbors $i \in V_1$ is above its threshold, i.e., $\sum_{i:(i,j)\in E} a_{ij}x_i \geq \nu_j$.

For RASC, let $\sigma(x)$ be a random variable representing the number of covered items in V_2 for a given x. It is known that $\sigma(x)(\omega)$ is submodular for $\omega \in \Omega$ [104]. Here, given an integer k and $\alpha \in (0,1]$, we rewrite problem (4.4) as

$$\max \quad \text{CVaR}_\alpha(\sigma(x)) \tag{4.16a}$$

$$\text{s.t.} \quad \sum_{i\in V_1} x_i \leq k \tag{4.16b}$$

$$x \in \mathbb{B}^n, \tag{4.16c}$$

where $\mathcal{X}$ is equivalent to $\{\sum_{i\in V_1} x_i \leq k, x \in \mathbb{B}^n\}$.

Under the true probability space, RASC admits an efficient CVaR oracle to solve problem (4.16). We propose an oracle that computes the function $\text{CVaR}_\alpha(\sigma(x))$ of problem (4.16) for $x \in \mathcal{X}$. Let $P(x,i)$ be the probability that a given solution x covers node $i \in V_2$. For the linear threshold model, $P(x,i) = \sum_{j\in V_1} a_{j,i}x_j$, and for the independent probability coverage model, $P(x,i) = 1 - \prod_{j\in V_1}(1 - a_{j,i}x_j)$. For a given x, the probability of covering exactly b nodes in V_2 out of a total of $|V_2| = m$ is represented as $\mathbb{P}(\sigma(x) = b)$. Note that $\mathbb{P}(\sigma(x) = b)$ is equal to the probability mass function of the Poisson binomial distribution [35, 84, 102], which is the discrete probability distribution of b successes in m Bernoulli trials, where each Bernoulli trial has a unique success probability. Let $\mathbb{P}_i(\sigma(x) = b)$ denote the probability of having b covered nodes in $V_2 \setminus \{i\}$ for a given x. Samuels [84] provides a formula to obtain

the value of probability mass function of the Poisson binomial distribution:

$$\mathbb{P}(\sigma(x) = b) = P(x, i) \times \mathbb{P}_i(\sigma(x) = b - 1) + (1 - P(x, i)) \times \mathbb{P}_i(\sigma(x) = b). \tag{4.17}$$

Next, we describe a dynamic program (DP) to compute $\mathrm{CVaR}_\alpha(\sigma(x))$ exactly [11, 110]. Let $V^i = \{1, \ldots, i\} \in V_2$ be the set of the first i nodes of V_2. Also let $A(x, i, j)$ represent the probability that the selection x covers j nodes among V^i for $0 \leq j \leq i, i \in V_2$. The DP recursion for $A(x, i, j)$ for $1 \leq j \leq i, i \in V_2$ is formulated as

$$A(x, i, j) = \begin{cases} A(x, i-1, j) \times (1 - P(x, i)), & j = 0 \\ A(x, i-1, j) \times (1 - P(x, i)) + A(x, i-1, j-1) \times P(x, i), & 0 < j < i \\ A(x, i-1, j-1) \times P(x, i), & j = i, \end{cases}$$

where the boundary condition is $A(x, 0, 0) = 1$. Note that $\mathrm{VaR}_\alpha(\sigma(x)) = \min\{j \in \mathbb{Z}_+ : \sum_{i=0}^{j} A(x, m, j) \geq \alpha\}$. The goal function $\mathrm{CVaR}_\alpha(\sigma(x))$ is equal to $\frac{1}{\alpha} \times \big(\sum_{j=0}^{\mathrm{VaR}_\alpha(\sigma(x))-1} A(x, m, j) \times j + \mathrm{VaR}_\alpha(\sigma(x)) \times (\alpha - \sum_{j=0}^{\mathrm{VaR}_\alpha(\sigma(x))-1} A(x, m, j)) \big)$. For a given x, the running time of the DP is $\mathcal{O}(nm + m^2)$, because obtaining $P(x, j)$ for all $j \in V_2$ is $\mathcal{O}(nm)$, and computing the recursion is $\mathcal{O}(m^2)$. Based on the CVaR oracle described above, given an integer k and $\alpha \in (0, 1]$, Algorithm 7 provides an optimal solution of problem (4.16) without sampling.

Under the assumption of a finite probability space, we generate a live-arc graph for each scenario, where the definition of live-arc graph is in [38] and [104]. For each scenario $\omega_i \in \Omega$, let $\sigma_i(x)$ denote the number of nodes covered in V_2 by the selection x for the live-arc graph $G_{\omega_i} = (V \cup V_2, E_{\omega_i})$. Let $y_j^i = 1$ if $j \in V_2$ is covered by the subset selection x for G_{ω_i}. Given a set of N scenarios, an integer k and $\alpha \in (0, 1]$, a deterministic equivalent formulation is

$$\max \quad \eta - \frac{1}{\alpha} \sum_{i \in [N]} p_i w_i \tag{4.18a}$$

$$\text{s.t.} \quad w_i \geq \eta - \sum_{j \in V_2} y_j^i \qquad \forall i \in [N] \tag{4.18b}$$

$$\sum_{d \in V_1} t_{jd}^i x_d \geq y_j^i \qquad \forall j \in V_2, \forall i \in [N] \tag{4.18c}$$

$$\sum_{i \in V_1} x_i \leq k \tag{4.18d}$$

$$x \in \mathbb{B}^n, y \in \mathbb{B}^{mN}, w \in \mathbb{R}_+^N, \eta \in \mathbb{R}, \tag{4.18e}$$

where $t_{jd}^i = 1$ if $j \in S_d$ for $\omega_i \in \Omega, d \in V_1$; otherwise, $t_{jd}^i = 0$ for all $j \in V_2 \setminus S_d, d \in V_1, \omega_i \in \Omega$. We can use state-of-the-art software package to solve formulation (4.18) directly; however, if it is large scale, we can also solve problem (4.16) by using Algorithm 9.

4.4 Computational Experiments

In this section, we report our experiments to demonstrate the effectiveness of our proposed methods. All algorithms are implemented in C++ with IBM ILOG CPLEX 12.7 Optimizer. All experiments were executed on a Windows 8.1 operating system with an Intel Core i5-4200U 1.60 GHz CPU, 8 GB DRAM, and x64 based processor. For the master problem of the decomposition algorithms and the deterministic mixed integer programming models, we specify the MIP search method as traditional branch-and-cut with the lazycallback function of CPLEX. We set the number of threads to one. CPLEX presolve process is turned off for the traditional branch-and-cut for solving the decomposition algorithms. The relative MIP gap tolerance of CPLEX is set to the default value, so a feasible solution which has an optimality gap of $10^{-4}\%$ is considered optimal. The time limit is set to 1800 seconds.

We focus on RASC under the independent probability coverage model in the computational study. We generate a complete bipartite graph with arcs from all nodes $i \in V_1$ to all $j \in V_2$, where each arc (i, j) is assigned an independent probability a_{ij} of being live. Suppose that we have two collection of sets V_1^1 and V_1^2, where each node $i \in V_1^1$ can cover a higher expected number of items than each node $j \in V_1^2$. In particular, we let $a_{ij} = 0.18 + i \times (0.22 - 0.18)/|V_1^1|$ for each $i \in V_1^1$, and $a_{ij} = (i - |V_1^1|) \times (0.04)/|V_1^2|$ for each $i \in V_1^2$, where we let $V_1^2 = \{|V_1^1| + 1, \ldots, n\}$. Let $\mathcal{V} = V_1 \cup V_2$. The size of bipartite graphs is $|\mathcal{V}| \in \{50, 100, 150\}$. Unless otherwise noted, we let $n = m = |\mathcal{V}|/2$ and $n = |V_1^1| + |V_1^2|$. We let $|V_1^1| = 10$ for all instances, and $|V_1^2| = n - 10$. We set up $k \in \{3, 5\}$ and $\alpha \in \{0.025, 0.05\}$.

We first demonstrate Algorithm 7 with three valid inequalities, the L-shape cut of [44], inequality (4.6) and lifted inequality (4.11). Note that for a given incumbent solution, $\bar{x}$,

which is a characteristic vector of the set $\bar{X}$, we consider the L-shape cut

$$\psi \leq \text{CVaR}_\alpha(\sigma(\bar{x})) + \sum_{j \in V \setminus \bar{X}} \big(\text{CVaR}_1(\sigma(V_1)) - \text{CVaR}_\alpha(\sigma(\bar{x}))\big) x_j. \tag{4.19}$$

The result is shown in Table 4.1. We refer to Algorithm 7 as "Oracle" in this subsection. Column "Oracle-LShape" denotes Algorithm 7 with the L-shape cut of [44]. Column "Oracle-Ineq (4.6)" denotes Algorithm 7 with inequality (4.6). Column "Oracle-Ineq (4.11)" denotes Algorithm 7 with inequality (4.11). Column "Oracle-Ineq (4.6) and (4.11)" denotes Algorithm 7 with both inequalities (4.6) and (4.11). Column "Time" denotes the total time of solving each instance for the master problem, in seconds. Column "Cuts" denotes the total number of user cuts added to the master problem. Column "Nodes" denotes the number of branch-and-bound nodes traced in the master problem.

We compare the performance of Oracle-LShape, Oracle-Ineq (4.6) and Oracle-Ineq (4.11) in Table 4.1. Table 4.1 shows that the solution time increases as k and $|\mathcal{V}|$ increase. We observe that Oracle-LShape cannot solve most instances within the time limit. Oracle-Ineq (4.6) and Oracle-Ineq (4.11) are faster than Oracle-LShape for the instances that are solvable by Oracle-LShape within the time limit. In addition, Oracle-Ineq generates a fewer number of optimality cuts and traces a fewer number of nodes compared to Oracle-LShape.

For the instances with $|\mathcal{V}| = 50$, Oracle-Ineq (4.11) is faster than Oracle-Ineq (4.6). For most of the instances with $|\mathcal{V}| \leq 100$, Oracle-Ineq (4.11) generates a fewer number of optimality cuts and traces a fewer number of nodes compared to Oracle-Ineq (4.6). We note that for the instances with a larger $|\mathcal{V}| = 150$, the solution time of Oracle-Ineq (4.11) is less than Oracle-Ineq (4.6). Recall that for a given incumbent solution $\bar{x}$, inequality (4.11) is generated by Algorithm 8. In Algorithm 8, we observe that for a given $\bar{x}$ and $1 \leq i \leq i' \leq r$, the following relation holds

$$\bar{\delta}_{j_i}(\bar{x}) \leq \bar{\delta}_{j_i'}(\bar{x}) \leq \text{CVaR}_\alpha(\sigma(V_1)).$$

From the above relation, if the size of $|\mathcal{V}|$ is large, then there may exist a large number of nodes with a high value of the lifting function $\delta_{j_i}(\bar{x})$, which is close to $\text{CVaR}_\alpha(\sigma(V_1))$. This

is the main reason why Oracle-Ineq (4.11) cannot perform well in instances with a large $|\mathcal{V}|$. To improve the performance of the computation, we add both inequalities (4.6) and (4.11) at each iteration in Algorithm 8. In Table 3.4, we observe that for the instances that are not solvable by both Oracle-Ineq (4.11) and Oracle-Ineq (4.6) within 100 seconds, the solution time of Oracle-Ineq (4.6) and (4.11) is shorter. Furthermore, for a hard instance $(|V|, \alpha, k) = (150, 0.05, 5)$, only Oracle-Ineq (4.6) and (4.11) can provide the optimal solution within the time limit.

Next, we demonstrate the sampling-based approach on RASC. We compare the deterministic equivalent formulation (4.18) referred to as (DEP) and Algorithm 9 with the submodular inequality (4.15) (referred to as DCG-Sub). We generate $|\Omega| = 1000$ equiprobable scenarios by using the live-arc graph scenario generation method we described earlier. For the sampling-based approach, we consider instances with larger network sizes, where $|\mathcal{V}| \in \{50, 100, 150, 200, 250, 300\}$. For each combination of $(\mathcal{V}, \alpha, k)$, we create three replications of the scenario set and report the average statistics. The result is shown in Table 4.2. Column "Time(u)" denotes the solution time in seconds and notation "(u)" denotes the number of unsolved instances out of the three instances tested for the corresponding setting.

We observe that solving the sample approximation problem with delayed cut generation (DCG) is more effective than solving the corresponding DEP model. In Table 4.2, DCG-Sub is faster than DEP, and DEP cannot solve instances with $|\mathcal{V}| \geq 150$ within the time limit. For most of the instances with $|\mathcal{V}| \geq 250$, DEP cannot even solve the LP relaxation within the time limit. Moreover, for the instances that are solvable by DEP within the time limit, DCG-Sub traces a fewer number of branch and bound nodes compared to DEP.

Note that although the sampling-based approach can solve instances with larger network sizes, the optimal solution provided by the sampling-based approach is an approximation of the optimal solution. We compare Table 4.1 and Table 4.2. We observe that for instances with $|\mathcal{V}| = 50$, the exact method using the true distribution can solve the instances faster than the sampling-based approach that approximates the true problem with a certain number of scenarios. For instance, for the setting with $|\mathcal{V}| = 50, \alpha = 0.05, k = 5$, the solution time of

Table 4.1: Algorithm 7 with different inequalities

			Oracle-LShape			Oracle-Ineq (4.6)			Oracle-Ineq (4.11)			Oracle-Ineq (4.6) and (4.11)		
$\lvert\mathcal{V}\rvert$	α	k	Time	Cuts	Nodes	Time	Cuts	Nodes	Time	Cuts	Nodes	Time	Cuts	Nodes
50	0.025	3	9	2315	2363	3	801	1915	≤ 1	224	336	≤ 1	398	667
50	0.025	5	≥ 1800	33988	36405	681	16683	52677	3	726	1960	5	1462	2264
50	0.05	3	10	2319	2363	2	799	1734	≤ 1	225	366	≤ 1	397	787
50	0.05	5	≥ 1800	32548	36569	492	14978	43320	3	713	1859	5	1484	2095
100	0.025	3	1152	19661	20300	11	1845	4287	24	1511	2321	15	1816	1252
100	0.025	5	≥ 1800	21834	53054	1154	17741	97510	825	13156	46789	560	15648	45091
100	0.05	3	1205	19634	20685	9	1486	4476	24	1542	2400	15	1808	1224
100	0.05	5	≥ 1800	21980	52479	505	11408	74808	826	12911	47823	238	10684	21191
150	0.025	3	≥ 1800	18922	22045	25	1832	12753	457	6775	10868	91	3046	1389
150	0.025	5	≥ 1800	19364	43796	1640	17781	109795	≥ 1800	13468	72593	1603	20278	92923
150	0.05	3	≥ 1800	20394	21649	6	1021	1115	445	6705	10815	59	1802	8489
150	0.05	5	≥ 1800	9484	135762	≥ 1800	14185	65856	≥ 1800	14759	68158	1582	19978	75957

Oracle-Ineq (4.6) and (4.11) is 5 seconds, but the average solution time with the sampling-based method DCG-Sub is 123 seconds for 1000 scenarios. These results show that, for smaller networks, the proposed exact method may be able to solve the problem to true optimality more efficiently than a sampling-based method, which is not able to guarantee optimality.

Table 4.2: A comparison between DCG-Sub and DEP.

$\|\mathcal{V}\|$	α	k	DCG-Sub			DEP	
			Time(u)	Cuts	Nodes	Time(u)	Nodes
50	0.025	3	10	47666	218	331	1845
50	0.025	5	62	13233	488	216	1409
50	0.05	3	22	67666	234	516	2809
50	0.05	5	123	17900	505	338	3015
100	0.025	3	33	109000	239	(3)	3298
100	0.025	5	220	232667	558	1508	3290
100	0.05	3	47	116333	238	1346	2075
100	0.05	5	274	230666	548	1293	2174
150	0.025	3	42	120000	238	(3)	1851
150	0.025	5	290	234000	590	(3)	1426
150	0.05	3	58	123000	239	(3)	2604
150	0.05	5	342	223667	566	1180(2)	516
200	0.025	3	52	120667	239	(3)	339
200	0.025	5	350	206633	648	(3)	52
200	0.05	3	65	122333	238	(3)	27
200	0.05	5	457	225667	598	(3)	99
250	0.025	3	56	123333	238	(3)	≤ 1
250	0.025	5	488	215333	653	(3)	3
250	0.05	3	71	122667	239	(3)	≤ 1
250	0.05	5	619	226000	623	(3)	≤ 1
300	0.025	3	66	124000	238	(3)	≤ 1
300	0.025	5	617	218667	670	(3)	≤ 1
300	0.05	3	83	123333	239	(3)	≤ 1
300	0.05	5	777	224333	637	(3)	≤ 1

Chapter 5

CONCLUSIONS AND FUTURE WORK

In this dissertation, we investigate mathematical programming models and methods to solve stochastic combinatorial optimization problems. We start, in Chapter 2, with a method to solve risk-neutral two-stage stochastic submodular optimization problems. Then, in Chapter 3, we introduce the concept of a probability oracle for a class of combinatorial chance-constrained programs and propose general methods to solve such CCPs. Finally, in Chapter 4, we introduce the concept of a conditional value-at-risk oracle for a class of risk-averse submodular maximization problems and propose general methods to solve this problem class. Next, we summarize the work presented in Chapters 2, 3, 4 and future research directions.

Chapter 2 is based on [105]. We consider a two-stage stochastic optimization model that exploits the submodularity of the objective function. A delayed constraint generation algorithm is proposed to solve such a model. We generalize the proposed algorithm to solve any two-stage stochastic program, where the second-stage value function is submodular. We apply our methods to an influence maximization problem arising in social networks. We show that exploiting the submodularity of the influence function leads to strong optimality cuts. Furthermore, our computational experiments with large-scale real-world test instances indicate that the algorithm performs favorably against the basic greedy heuristic of [38] for this problem. In most instances, our algorithm finds a solution with provable optimality guarantees more quickly than a basic implementation of the greedy heuristic, which can only provide a 0.63 performance guarantee. Our algorithm is applicable to many other variants of the influence maximization problem for which the influence function is submodular.

Our results on optimization-based methods for the fundamental influence maximization problems provide a foundation to build algorithms for more advanced models, such as the

adaptive model of [88], where a subset of additional seed nodes is selected in the second stage based on the realization of some of the uncertain parameters and the seed nodes selected in the first stage. The decomposition methods of [89, 31] and [108] can be employed in this case to convexify the second stage problems that involve binary decisions. Another possible future research direction is to develop optimization-based methods for the problem of marketing to nodes [38, 39] to increase their probabilities of getting activated.

Chapter 3 is based on [104]. We propose a general delayed cut generation method to solve chance-constrained combinatorial optimization problems exactly (without sampling) when there is an efficient oracle to check whether a given solution satisfies the chance constraint. In addition, we show that the oracle can be used as a detector for checking the feasibility of the solution given by a sampling-based approach. We demonstrate our proposed methods on a probabilistic partial set covering problem (PPSC) considered in the social networks literature, under certain probability distributions, one of which is finite but exponential (independent probability coverage) and the other is a continuous distribution (linear threshold). For the linear threshold formulation, we give a compact MIP that linearly encodes the probability oracle within the optimization model. For PPSC, we give strong valid inequalities for the deterministic equivalent formulation of the sample approximation problem and show that the proposed inequalities subsume the submodular inequalities that are valid for this problem. In our computational study of the proposed methods, we observe that the exact method is preferred for small networks. It provides provably optimal solutions with respect to the true distribution efficiently. However, we see that the sampling-based methods scale better when the size of the problem increases if they are able to exploit the problem structure. In particular, we show that using the proposed valid inequalities in a branch-and-bound framework enables the solution of problems with larger network sizes. While the optimal solution to the sample approximation problem may not even be feasible, our oracle-based method can check and correct the feasibility of the solution to obtain a high-quality feasible solution to the original problem. We note that our methods are generally applicable to other problems with the desired structure. For example, the probabilistic set covering problem

with a circular distribution considered in [14, 55] fits into our framework, although, in this case, we can also provide a compact MIP using the formulable structure of the probability oracle.

In Chapter 3, we consider a class of CCPs with binary decision variables. A possible direction is to use the idea of oracles to solve other classes of CCPs exactly, such as those with continuous decision variables, in which case we are not able to use the no-good cuts. In addition, it will be useful to exploit the structure of the problems to derive more effective feasibility cuts for the exact algorithm.

In Chapter 4, we consider risk-averse solutions to a class of stochastic submodular maximization problems that optimizes the conditional value-at-risk (CVaR) of a random objective function at a given risk level, where the random objective function is submodular. We propose a general method to solve risk-averse submodular maximization problems (RASM) exactly (without sampling) when there is an efficient oracle to compute the CVaR of the random objective function exactly. Moreover, under the assumption of a finite probability space, we provide a two-stage stochastic programming model and another algorithm to solve RASM. We demonstrate our proposed methods on a risk-averse set covering problem (RASC) under probability distributions. We show that the exact method is preferred for small networks, and the sampling-based approach can solve instances with larger network sizes. In future work, the strength of optimality cuts for both the exact method and the sampling-based approach can be improved.

BIBLIOGRAPHY

[1] A. Abdi and R. Fukasawa. On the mixing set with a knapsack constraint. *Mathematical Programming*, 157(1):191–217, 2016.

[2] W. P. Adams and H. D. Sherali. Mixed-integer bilinear programming problems. *Mathematical Programming*, 59(3):279–305, 1993.

[3] S. Ahmed. Convexity and decomposition of mean-risk stochastic programs. *Mathematical Programming*, 106:433–446, 2006.

[4] S. Ahmed and A. Atamtürk. Maximizing a class of submodular utility functions. *Mathematical Programming*, 128(1):149–169, 2011.

[5] S. Ahmed and D. J. Papageorgiou. Probabilistic set covering with correlations. *Operations Research*, 61(2):438–452, 2013.

[6] R. K. Ahuja, T. L. Magnanti, and J. B. Orlin. *Network Flows: Theory, Algorithms, and Applications*. Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 1993.

[7] P. Artzner, F. Delbaen, J. Eber, and D. Heath. Coherent measures of risk. *Mathematical Finance*, 9(3):203–228, 1999.

[8] A. Arulselvan, C. W. Commander, L. Elefteriadou, and P. M. Pardalos. Detecting critical nodes in sparse graphs. *Computers and Operations Research*, 36(7):2193 – 2200, 2009.

[9] E. Balas. A class of location, distribution and scheduling problems: Modeling and solution methods. Technical Report, Design Research Center, Carnegie Mellon University, Pittsburgh, PA, 1982.

[10] B. Balasundaram, S. Butenko, and I. V. Hicks. Clique relaxations in social network analysis: The maximum k-plex problem. *Operations Research*, 59(1):133–142, 2011.

[11] R. E. Barlow and K. D. Heidtmann. Computing k-out-of-n system reliability. *IEEE Transactions on Reliability*, 33(4):322–323, 1984.

[12] J. Benders. Partitioning procedures for solving mixed-variables programming problems. *Numerische Mathematik*, 4(1):238–252, 1962.

[13] P. Beraldi and M. E. Bruni. An exact approach for solving integer problems under probabilistic constraints with random technology matrix. *Annals of Operations Research*, 177(1):127–137, 2010.

[14] P. Beraldi and A. Ruszczyński. The probabilistic set-covering problem. *Operations Research*, 50(6):956–967, 2002.

[15] D. Bertsimas and J. Tsitsiklis. *Introduction to Linear Optimization*. Athena Scientific, 1st edition, 1997.

[16] K. Bimpikis, A. Ozdaglar, and E. Yildiz. Competitive targeted advertising over networks. *Operations Research*, 64(3):705–720, 2016.

[17] J. R. Birge and F. Louveaux. *Introduction to stochastic programming*. Springer Verlag, New York, 1997.

[18] C. Borgs, M. Brautbar, J. Chayes, and B. Lucier. Maximizing social influence in nearly optimal time. In *Proceedings of the Twenty-Fifth Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '14, pages 946–957. SIAM, 2014.

[19] A. Charnes, W. W. Cooper, and G. H. Symonds. Cost horizons and certainty equivalents: An approach to stochastic programming of heating oil. *Management Science*, 4(3):235–263, 1958.

[20] W. Chen, L. V. Lakshmanan, and C. Castillo. Information and influence propagation in social networks. *Synthesis Lectures on Data Management*, 5(4):1–177, 2013.

[21] W. Chen, W. Lu, and N. Zhang. Time-critical influence maximization in social networks with time-delayed diffusion process. In *Proceedings of the 26th AAAI Conference on Artificial Intelligence*, 2012.

[22] W. Chen, Y. Wang, and S. Yang. Efficient influence maximization in social networks. In *Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '09, pages 199–208, New York, NY, USA, 2009. ACM.

[23] R. Church and C. R. Velle. The maximal covering location problem. *Papers in Regional Science*, 32(1):101–118, 1974.

[24] R. Cohen and L. Katzir. The generalized maximum coverage problem. *Information Processing Letters*, 108(1):15–22, 2008.

[25] I. Contreras and E. Fernández. Hub location as the minimization of a supermodular set function. *Operations Research*, 62(3):557–570, 2014.

[26] G. Cornuéjols, M. L. Fisher, and G. L. Nemhauser. Location of bank accounts to optimize float: An analytic study of exact and approximate algorithms. *Management Science*, 23(8):789–810, 1977.

[27] P. Domingos and M. Richardson. Mining the network value of customers. In *Proceedings of the Seventh ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '01, pages 57–66, New York, NY, USA, 2001. ACM.

[28] G. Ergün. Human sexual contact network as a bipartite graph. *Physica A*, 308:483–488, 2002.

[29] Z. Ertem, A. Veremyev, and S. Butenko. Detecting large cohesive subgroups with high clustering coefficients in social networks. *Social Networks*, 46:1 – 10, 2016.

[30] M. Fischetti and M. Monaci. Cutting plane versus compact formulations for uncertain (integer) linear programs. *Mathematical Programming Computation*, 4(3):239–273, 2012.

[31] D. Gade, S. Küçükyavuz, and S. Sen. Decomposition algorithms with parametric Gomory cuts for two-stage stochastic integer programs. *Mathematical Programming*, 144(1–2):39–64, 2014.

[32] D. Günneç, S. Raghavan, and R. Zhang. Tailored incentives and least cost influence maximization on social networks. *Technical report*, 2016.

[33] P. Hines, K. Balasubramaniam, and E. Sanchez. Cascading failures in power grids. *IEEE Potentials*, 28(5):24–30, September 2009.

[34] D. S. Hochbaum. Approximation algorithms for np-hard problems. *PWS Publishing*, 1996.

[35] W. Hoeffding. On the distribution of the number of successes in independent trials. *Ann. Math. Statist.*, 27(3):713–721, 1956.

[36] R. M. Karp. Reducibility among combinatorial problems. *Complexity of Computer Computations*, pages 85–103, 1972.

[37] Y. Kawahara, K. Nagano, K. Tsuda, and J. A. Bilmes. Submodularity cuts and applications. In *Proceedings of the 22nd International Conference on Neural Information Processing Systems*, NIPS '09, pages 916–924, 2009.

[38] D. Kempe, J. Kleinberg, and É. Tardos. Maximizing the spread of influence through a social network. In *Proceedings of the Ninth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '03, pages 137–146, New York, NY, USA, 2003. ACM.

[39] D. Kempe, J. Kleinberg, and É. Tardos. Maximizing the spread of influence through a social network. *Theory of Computing*, 11(4):105–147, 2015.

[40] S. Khuller, A. Moss, and J. Naor. The budgeted maximum coverage problem. *Information Processing Letters*, 70(1):39–45, 1999.

[41] B. Klimt and Y. Yang. Introducing the Enron corpus. In *First Conference on Email and Anti-Spam*, CEAS '04, 2004.

[42] S. Küçükyavuz. On mixing sets arising in chance-constrained programming. *Mathematical Programming*, 132(1):31–56, 2012.

[43] S. Küçükyavuz and S. Sen. An introduction to two-stage stochastic mixed-integer programming. *INFORMS TutORials in Operations Research*, pages 1–27, 2017.

[44] G. Laporte and F. Louveaux. The integer L-shaped method for stochastic integer programs with complete recourse. *Operations Research Letters*, 13(3):133–142, 1993.

[45] H. Lee, G. L. Nemhauser, and Y. Wang. Maximizing a submodular function by integer programming: Polyhedral results for the quadratic case. *European Journal of Operational Research*, 94(1):154 – 166, 1996.

[46] M. Lejeune. Pattern-based modeling and solution of probabilistically constrained optimization problems. *Operations Research*, 60(6):1356–1372, 2012.

[47] J. Leskovec, J. Kleinberg, and C. Faloutsos. Graph evolution: Densification and shrinking diameters. *ACM Transactions on Knowledge Discovery from Data*, 1(1):2, 2007.

[48] J. Leskovec, A. Krause, C. Guestrin, C. Faloutsos, J. VanBriesen, and N. Glance. Cost-effective outbreak detection in networks. In *Proceedings of the 13th ACM SIGKDD international conference on Knowledge discovery and data mining*, KDD '07, pages 420–429, New York, NY, USA, 2007. ACM.

[49] J. Leskovec, K. J. Lang, A. Dasgupta, and M. W. Mahoney. Community structure in large networks: Natural cluster sizes and the absence of large well-defined clusters. *Internet Mathematics*, 6(1):29–123, 2009.

[50] B. Liu, G. Cong, D. Xu, and Y. Zeng. Time constrained influence maximization in social networks. In *2012 IEEE 12th International Conference on Data Mining (ICDM)*, pages 439–448, Dec 2012.

[51] X. Liu, F. Kilinc-Karzan, and S. Küçükyavuz. On intersection of two mixing sets with applications to joint chance-constrained programs. *Mathematical Programming*, 2017.

[52] X. Liu and S. Küçükyavuz. A polyhedral study of the static probabilistic lot-sizing problem. *Annals of Operations Research*, 261(1-2):233–254, 2018.

[53] X. Liu, S. Küçükyavuz, and J. Luedtke. Decomposition algorithms for two-stage chance-constrained programs. *Mathematical Programming*, 157(1):219–243, 2016.

[54] J. Luedtke. A branch-and-cut decomposition algorithm for solving chance-constrained mathematical programs with finite support. *Mathematical Programming*, 146(1):219–244, 2014.

[55] J. Luedtke and S. Ahmed. A sample approximation approach for optimization with probabilistic constraints. *SIAM Journal on Optimization*, 19(2):674–699, 2008.

[56] J. Luedtke, S. Ahmed, and G. L. Nemhauser. An integer programming approach for linear programs with probabilistic constraints. *Mathematical Programming*, 122(2):247–272, 2010.

[57] N. Madar, T. Kalisky, R. Cohen, D. Ben-Avraham, and S. Havlin. Immunization and epidemic dynamics in complex networks. *The European Physical Journal B - Condensed Matter and Complex Systems*, 38(2):269–276, 2004.

[58] T. Maehara. Risk averse submodular utility maximization. *Operations Research Letters*, 43:526–529, 2015.

[59] T. L. Magnanti and R. T. Wong. Accelerating benders decomposition: Algorithmic enhancement and model selection criteria. *Operations Research*, 29(3):464–484, 1981.

[60] G. McCormick. Computability of global solutions to factorable nonconvex programs: Part I – convex underestimating problems. *Mathematical Programming*, 10(1):147–175, 1976.

[61] N. Miller and A. Ruszczyński. Risk-averse two-stage stochastic linear programming: Modeling and decomposition. *Operations Research*, 59:125–132, 2011.

[62] E. Mossel and S. Roch. On the submodularity of influence in social networks. In *Proceedings of the Thirty-ninth Annual ACM Symposium on Theory of Computing*, STOC '07, pages 128–134, New York, NY, USA, 2007. ACM.

[63] E. Mossel and S. Roch. Submodularity of influence in social networks: From local to global. *SIAM Journal on Computing*, 39(6):2176–2188, 2010.

[64] G. Nemhauser and L. Wolsey. Maximizing submodular set functions: Formulations and analysis of algorithms. In P. Hansen, editor, *Annals of Discrete Mathematics (11) Studies on Graphs and Discrete Programming*, volume 59 of *North-Holland Mathematics Studies*, pages 279 – 301. North-Holland Mathematics Studies, 1981.

[65] G. Nemhauser, L. Wolsey, and M. Fisher. An analysis of approximations for maximizing submodular set functions—I. *Mathematical Programming*, 14(1):265–294, 1978.

[66] G. L. Nemhauser and L. A. Wolsey. *Integer and Combinatorial Optimization*. Wiley-Interscience, New York, NY, USA, 1988.

[67] A. Nemirovski and A. Shapiro. Scenario approximations of chance constraints. *Probabilistic and Randomized Methods for Design under Uncertainty*, pages 3–47, 2006.

[68] M. E. J. Newman. The structure and function of complex networks. *SIAM Rev*, 45(2):167–256, 2003.

[69] N. Noyan. Risk-averse two-stage stochastic programming with an application to disaster management. *Computers and Operations Research*, 39:365–386, 2012.

[70] F. Oliveira, I. Grossmann, and S. Hamacher. Accelerating benders stochastic decomposition for the optimization under uncertainty of the petroleum product supply chain. *Computers & Operations Research*, 49:47 – 58, 2014.

[71] T. Opsahl and P. Panzarasa. Clustering in weighted networks. *Social Networks*, 31(2):155–163, 2009.

[72] A. Ostfeld and E. Salomons. Optimal layout of early warning detection stations for water distribution systems security. *Journal of Water Resources Planning and Management*, 130(5):377–385, 2004.

[73] N. Papadakos. Practical enhancements to the Magnanti-Wong method. *Operations Research Letters*, 36(4):444 – 449, 2008.

[74] C. H. Papadimitriou and K. Steiglitz. *Combinatorial optimization: algorithms and complexity*. Prentice-Hall, 1982.

[75] D. Pollard. *A User's Guide to Measure Theoretic Probability*. Cambridge University Press, Cambridge, UK, 2001.

[76] A. Prékopa. Contributions to the theory of stochastic programming. *Mathematical Programming*, 4(1):202–221, 1973.

[77] A. Prékopa. Dual method for the solution of a one-stage stochastic programming problem with random RHS obeying a discrete probability distribution. *ZOR - Methods and Models of Operations Research*, 34(6):441–461, 1990.

[78] S. Raghavan and R. Zhang. Weighted target set selection on social networks. *Technical report*, 2015.

[79] J. R.Birge and F. V.Louveaux. A multicut algorithm for two-stage stochastic linear programs. *European Journal of Operational Research*, 34(3):384–392, 1988.

[80] M. Ripeanu, I. Foster, and A. Iamnitchi. Mapping the gnutella network: Properties of large-scale peer-to-peer systems and implications for system design. *IEEE Internet Computing*, 6(1):50–57, 2002.

[81] R. T. Rockafellar and S. Uryasev. Optimization of conditional value-at-risk. *Journal of Risk*, 2:21–41, 2000.

[82] R. T. Rockafellar and S. Uryasev. Conditional value-at-risk for general loss distributions. *Journal of Banking and Finance*, 26:1443–1471, 2002.

[83] A. Ruszczyński. Probabilistic programming with discrete distributions and precedence constrained knapsack polyhedra. *Mathematical Programming*, 93(2):195–215, 2002.

[84] S. M. Samuels. On the number of successes in independent trials. *The Annals of Mathematical Statistics*, 36(4):1272–1278, 1965.

[85] T. Santoso, S. Ahmed, M. Goetschalckx, and A. Shapiro. A stochastic programming approach for supply chain network design under uncertainty. *European Journal of Operational Research*, 167(1):96 – 115, 2005.

[86] A. Saxena, V. Goyal, and M. A. Lejeune. MIP reformulations of the probabilistic set covering problem. *Mathematical Programming*, 121(1):1–31, 2010.

[87] R. Schultz and S. Tiedemann. Conditional value-at-risk in stochastic programs with mixed-integer recourse. *Mathematical Programming*, 105:365–386, 2006.

[88] L. Seeman and Y. Singer. Adaptive seeding in social networks. In *2013 IEEE 54th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 459–468, 2013.

[89] S. Sen. Stochastic mixed-integer programming algorithms: Beyond Benders' decomposition. In J. J. Cochran, L. A. Cox, P. Keskinocak, J. P. Kharoufeh, and J. C. Smith, editors, *Wiley Encyclopedia of Operations Research and Management Science*. John Wiley & Sons, Inc., 2010.

[90] A. Shapiro, D. Dentcheva, and A. Ruszczyński. *Lectures on Stochastic Programming: Modeling and Theory*. Society for Industrial Mathematics, Philadelphia, PA, USA, 2009.

[91] H. D. Sherali and B. J. Lunday. On generating maximal nondominated Benders cuts. *Annals of Operations Research*, 210(1):57–72, 2013.

[92] Y. Song and T. N. Dinh. Optimal containment of misinformation in social media: A scenario-based approach. In Z. Zhang, L. Wu, W. Xu, and D.-Z. Du, editors, *Combinatorial Optimization and Applications: 8th International Conference Proceedings, COCOA 2014, Wailea, Maui, HI, USA, December 19-21, 2014*, pages 547–556. Springer International Publishing, Cham, 2014.

[93] Y. Song, J. R. Luedtke, and S. Küçükyavuz. Chance-constrained binary packing problems. *INFORMS Journal on Computing*, 26(4):735–747, 2014.

[94] M. Sviridenko. A note on maximizing a submodular set function subject to a knapsack constraint. *Operations Research Letters*, 32(1):41 – 43, 2004.

[95] W. van Ackooij, A. Frangioni, and W. Oliveira. Inexact stabilized Benders' decomposition approaches with application to chance-constrained problems with finite support. *Computational Optimization and Applications*, 65(3):637–669, 2016.

[96] W. van Ackooij and C. Sagastizábal. Constrained bundle methods for upper inexact oracles with application to joint chance constrained energy problems. *SIAM Journal on Optimization*, 24(2):733–765, 2014.

[97] R. van der Hofstad. *Random Graphs and Complex Networks: Volume 1*. Cambridge University Press, Cambridge, UK, 2016.

[98] R. Van Slyke and R. Wets. L-shaped linear programs with applications to optimal control and stochastic programming. *SIAM Journal on Applied Mathematics*, 17(4):638–663, 1969.

[99] R. V. Vohra and N. G. Hall. A probabilistic analysis of the maximal covering location problem. *Discrete Applied Mathematics*, 43(2):175–183, 1993.

[100] S. W. Wallace. Investing in arcs in a network to maximize the expected max flow. *Networks*, 17(1):87–103, 1987.

[101] C. Wang, W. Chen, and Y. Wang. Scalable influence maximization for independent cascade model in large-scale social networks. *Data Mining and Knowledge Discovery*, 25(3):545–576, 2012.

[102] Y. H. Wang. On the number of successes in independent trials. *Statistica Sinica*, 3(2):295–312, 1993.

[103] R. D. Wollmer. Investments in stochastic maximum flow networks. *Annals of Operations Research*, 31(1):457–467, 1991.

[104] H. Wu and S. Küçükyavuz. *Chance-Constrained Combinatorial Optimization with a Probability Oracle and Its Application to Probabilistic Partial Set Covering.* Arxiv, 2017.

[105] H. Wu and S. Küçükyavuz. A two-stage stochastic programming approach for influence maximization in social networks. *Computational Optimization and Applications*, 69(3):563–595, 2018.

[106] P. Xanthopoulos, A. Arulselvan, V. Boginski, and P. Pardalos. A retrospective review of social networks. In *Social Network Analysis and Mining, 2009. ASONAM '09. International Conference on Advances in*, pages 300–305, 2009.

[107] J. Yu and S. Ahmed. Maximizing a class of submodular utility functions with constraints. *Mathematical Programming*, 162(1-2):145–164, 2017.

[108] M. Zhang and S. Küçükyavuz. Finitely convergent decomposition algorithms for two-stage stochastic pure integer programs. *SIAM Journal on Optimization*, 24(4):1933–1951, 2014.

[109] M. Zhang, S. Küçükyavuz, and S. Goel. A branch-and-cut method for dynamic decision making under joint chance constraints. *Management Science*, 60(5):1317–1333, 2014.

[110] P. Zhang, W. Chen, X. Sun, Y. Wang, and J. Zhang. Minimizing seed set selection with probabilistic coverage guarantee in a social network. In *Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '14, pages 1306–1315, New York, NY, USA, 2014. ACM.

[111] M. Zhao, K. Huang, and B. Zeng. A polyhedral study on chance constrained program with random right-hand side. *Mathematical Programming*, 166(1–2):19–64, 2017.

Appendix A

ALTERNATIVE BENDERS OPTIMALITY CUTS FOR LIVE-ARC GRAPH MODELS

In this section, we present optimality cuts that can be obtained by traditional methods. First, we give an explicit linear programming (LP) formulation for the subproblems (2.2) used to calculate $\sigma_\omega(x)$ for live-arc graph models such as independent cascade or linear threshold. Observe that the maximum number of nodes reachable from nodes X (corresponding to the decision vector x) in graph G_ω can be formulated as a maximum flow problem an a modified graph $G'_\omega = (V \cup \{s,t\}, A'_\omega)$, where s is the source node, t is the sink node, and A'_ω includes the arcs A_ω and arcs (s,i) and (i,t) for all $i \in V$. Let the capacity of the arcs (i,t), $i \in V$ be one, and the capacity of arcs $(i,j) \in A_\omega$ be n (the maximum flow possible on any arc). In addition, we would like the arcs $(s,i), i \in V$ to have a capacity of n if $x_i = 1$ and 0 otherwise. Therefore, we let the capacity of arc (s,i) be nx_i. The reader might wonder why we create an arc (s,i) if a node i is not activated. To see why, note that in a two-stage stochastic programming framework, we need to build a second-stage model that is correct for any first-stage decision x. It is easy to see that the maximum flow on this graph is equal to the maximum number of vertices reachable from the seeded nodes X. The LP formulation of the second-stage problem for scenario $\omega \in \Omega$ is

$$\sigma_\omega(x) = \quad \max \quad \sum_{i \in V} y_{si} \tag{A.1a}$$

$$\text{s.t.} \quad \sum_{j:(j,i)\in A'_\omega} y_{ji} - \sum_{j:(i,j)\in A'_\omega} y_{ij} = 0, \qquad i \in V \qquad (u_i^\omega) \tag{A.1b}$$

$$y_{si} \le nx_i, \qquad i \in V \qquad (v_{si}^\omega) \tag{A.1c}$$

$$y_{ij} \le n, \qquad (i,j) \in A_\omega \qquad (v_{ij}^\omega) \tag{A.1d}$$

$$y_{it} \leq 1, \qquad i \in V \qquad (v_{it}^{\omega}) \tag{A.1e}$$

$$y_{ij} \geq 0, \qquad (i,j) \in A'_{\omega}, \tag{A.1f}$$

where y_{ij} represents the flow on arc $(i,j) \in A'_{\omega}$, and the dual variables associated with each constraint are defined in parentheses. Note that the subproblems are feasible for any $\omega \in \Omega$ and $x \in \{0,1\}^n$ (we can always send zero flows), therefore this problem is said to have *complete recourse*. The dual of the second-stage problem (A.1) is

$$\sigma_{\omega}(x) = \quad \min \quad \sum_{i \in V} (n x_i v_{si}^{\omega} + v_{it}^{\omega}) + \sum_{(i,j) \in A_{\omega}} n v_{ij}^{\omega} \tag{A.2a}$$

$$\text{s.t.} \quad u_i^{\omega} + v_{si}^{\omega} \geq 1, \qquad i \in V \tag{A.2b}$$

$$u_j^{\omega} - u_i^{\omega} + v_{ij}^{\omega} \geq 0, \qquad (i,j) \in A_{\omega} \tag{A.2c}$$

$$-u_i^{\omega} + v_{it}^{\omega} \geq 0, \qquad i \in V \tag{A.2d}$$

$$v_{ij}^{\omega} \geq 0, \qquad (i,j) \in A'_{\omega}. \tag{A.2e}$$

Note that we can write a large-scale mixed-integer program, known as the deterministic equivalent program (DEP), to solve the independent cascade problem. To do this, we create copies of the second-stage variables y_{ij}^{ω} for all $\omega \in \Omega$, where y_{ij}^{ω} represents the flow on arc $(i,j) \in A'_{\omega}$ under scenario $\omega \in \Omega$. The DEP is formulated as

$$\max \quad \sum_{\omega \in \Omega} p_{\omega} \sum_{i \in V} y_{si}^{\omega} \tag{A.3a}$$

$$\text{s.t.} \quad \sum_{j \in V} x_j \leq k \tag{A.3b}$$

$$\sum_{j:(j,i) \in A'_{\omega}} y_{ji} - \sum_{j:(i,j) \in A'_{\omega}} y_{ij} = 0, \qquad \omega \in \Omega, i \in V \tag{A.3c}$$

$$y_{si}^{\omega} \leq n x_i, \qquad \omega \in \Omega, i \in V \tag{A.3d}$$

$$y_{ij}^{\omega} \leq n, \qquad \omega \in \Omega, (i,j) \in A_{\omega} \tag{A.3e}$$

$$y_{it}^{\omega} \leq 1, \qquad \omega \in \Omega, i \in V \tag{A.3f}$$

$$x \in \{0,1\}^n, y_{ij}^{\omega} \geq 0, \omega \in \Omega, (i,j) \in A'_{\omega}. \tag{A.3g}$$

It is well-established in the stochastic programming field that due to its large size, it is not practical to solve DEP directly. Instead, as is commonly done, we consider the use of Benders decomposition method [12, 98] utilizing the structure of this large-scale MIP.

A naive way of generating the optimality cuts is to solve the subproblem (A.1) for each $\omega \in \Omega$ as an LP (in line 5 of Algorithm 3) to obtain $\sigma_\omega(\bar{x})$, and the corresponding dual vector $(\bar{u}^\omega, \bar{v}^\omega)$. Then the optimality cut is

$$\theta_\omega \leq \sum_{i \in V} (n x_i \bar{v}^\omega_{si} + \bar{v}^\omega_{it}) + \sum_{(i,j) \in A_\omega} n \bar{v}^\omega_{ij}. \tag{A.4}$$

We refer to the optimality cuts (A.4) obtained by solving the subproblems as an LP as the *LP-based optimality cuts*.

Next, we discuss a more efficient way of obtaining the optimality cuts by utilizing the fact that the subproblems are maximum flow problems, which can be solved in polynomial time using specialized algorithms. In particular, for our problem, one only needs to solve a reachability problem to obtain the corresponding maximum flow. Reachability problem in a graph can be solved in linear time in the number of arcs using breadth- or depth-first search. We describe the equivalence of the maximum flow problem defining the evaluation of the influence spread to the graph reachability problem next.

For a given first-stage solution $\bar{x}$ and the corresponding seed set $\bar{X}$, let $\hat{R}(\bar{X}) \subseteq V$ be the set of nodes in V reachable from s, $R(\bar{X}) = \hat{R}(\bar{X}) \setminus \bar{X}$ be the set of nodes reachable from s not including the seed nodes $\bar{X}$, and $\bar{R}(\bar{X}) = V \setminus \hat{R}(\bar{X})$ be the set of nodes in V *not* reachable from s in G'_ω. From maximum flow minimum cut theorem (see, e.g., [6]), we can show that a minimum cut is given by $(\hat{R}(\bar{X}) \cup \{s\}, \bar{R}(\bar{X}) \cup \{t\})$. (See the maximum flow formulation of this problem for a given $\bar{X}$ and scenario $\omega \in \Omega$ in Figure A.1.) Let $u^\omega_i = 1$ if $i \in \hat{R}(\bar{X})$, and $u^\omega_i = 0$, if $i \in \bar{R}(\bar{X})$. In addition, for $(i,j) \in A'_\omega$, let $v^\omega_{ij} = 1$ if $i \in \hat{R}(\bar{X}) \cup \{s\}$ and $j \in \bar{R}(\bar{X}) \cup \{t\}$, otherwise let $v^\omega_{ij} = 0$. It is easy to check that this choice of the dual variables is feasible. Furthermore, this choice is optimal. To see this, note that the objective

value of the dual is

$$\sum_{i\in V}(nx_i\bar{v}^{\omega}_{si}+\bar{v}^{\omega}_{it})+\sum_{(i,j)\in A_\omega} n\bar{v}^{\omega}_{ij}=\sum_{i\in\bar{R}(\bar{X})} nx_i+\sum_{i\in\hat{R}(\bar{X})} 1+\sum_{(i,j)\in(\hat{R}(\bar{X}),\bar{R}(\bar{X}))} n=|\hat{R}(\bar{X})|,$$

because $x_i = 0$ for $i \in \bar{R}(\bar{X})$ and there can be no arc $(i, j) \in A_\omega$ with $i \in \hat{R}(\bar{X})$, $j \in \bar{R}(\bar{X})$ (otherwise j would be reachable from s and hence it will be in $\hat{R}(\bar{X})$). Because the optimal objective value of the primal subproblem is $\sigma_\omega(\bar{x}) = |\hat{R}(\bar{X})|$, this dual solution must be optimal. With this choice of the optimal dual vector, we obtain the Benders optimality cut

$$\theta_\omega \le \sigma_\omega(\bar{x}) + \sum_{i\in\bar{R}(\bar{X})} nx_i. \tag{A.5}$$

We refer to the optimality cuts (A.5) obtained by solving the subproblems as reachability problems as *combinatorial optimality cuts*. Wallace [100] and Wollmer [103] use the same type of optimality cuts for a problem of investing in arc capacities of a network to maximize flow under stochastic demands. This problem is a relaxation of our problem in that the first stage variables are continuous. Hence, submodular inequalities cannot be used in their problem context.

Note that inequality (A.5) can also be seen as a big-M type inequality. For $x = \bar{x}$, with the associated seed set $\bar{X}$, we get a correct upper bound on θ_ω as $\sigma_\omega(x)$. For any other $x \neq \bar{x}$, if $x_i = 1$ for some $i \in \bar{R}(\bar{X})$, then the upper bound on θ_ω given by inequality (A.5) is trivially valid, because $\sigma_\omega(x) \le n$ for any $x \in \{0, 1\}^n$. Finally, for any $x \neq \bar{x}$, if $x_i = 0$ for all $i \in \bar{R}(\bar{X})$, then we must have $x_j = 0$ for some $j \in \bar{X}$ and $x_\ell = 1$ from some $\ell \in R(\bar{X})$. However, because ℓ is reachable from $\bar{X}$, replacing j with ℓ will not increase the number of reachable nodes, i.e., $\sigma_\omega(x) \le \sigma_\omega(\bar{x})$. Therefore, inequality (A.5) is valid.

Magnanti and Wong [59] propose a method to strengthen Benders cuts in cases when the dual of the subproblems is degenerate (see also, [73, 91], for other enhancements of this method). The method chooses, among alternative dual optimal solutions to the subproblem, one that is not dominated. While this idea is useful to strengthen the weak Benders cut (A.4) (in particular, inequality (A.5) corresponding to one choice of optimal dual solutions),

we note that it alone cannot lead to the stronger cuts given by the submodular inequalities (2.7). To see this note, first, that all extreme points of the dual subproblem (A.2) are integral. So any non-integral dual feasible solution is a convex combination of these extreme points. Then note that an optimality cut of the form (A.4) obtained from the dual is non-dominated only if the corresponding dual solution is an extreme point (otherwise the optimality cut would be a convex combination of the optimality cuts corresponding to the extreme points). As a result, $\bar{v}^{\omega}_{si} \in \mathbb{Z}$ for all $i \in V$, hence submodular inequalities (2.7) cannot be expressed as inequalities (A.4) obtained from non-dominated extreme point optimal dual solutions to the subproblem (A.2).

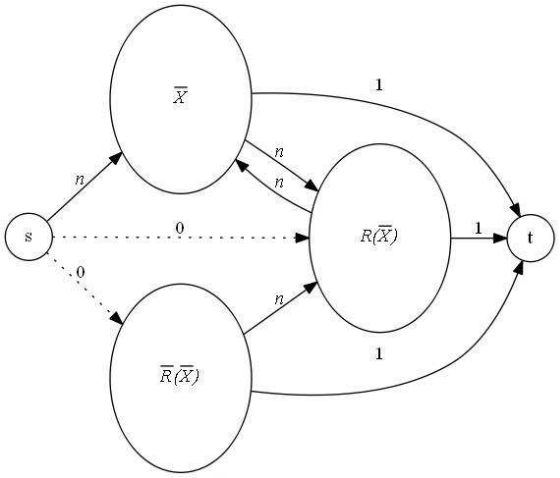

Figure A.1: Maximum flow formulation of the influence function.

Finally, note that because the first-stage problem is a pure binary optimization problem, one can also consider the optimality cuts proposed in the integer L-shaped method of [44]. The resulting inequality, for $\omega \in \Omega$ and a given $\bar{x}$, with an associated seed set $\bar{X}$, is

$$\theta_\omega \leq \sigma_\omega(\bar{x}) + \sum_{i \in V \setminus \bar{X}} (n - \sigma_\omega(\bar{x}))x_i. \tag{A.6}$$

This inequality can be strengthened by the same observation that replacing a node $j \in \bar{X}$ with a node $\ell \in R(\bar{X})$ does not increase the number of reachable nodes. Therefore, we can

reduce the coefficient of x_ℓ in inequality (A.6) to obtain a strengthened version of the integer L-shaped optimality cut (A.6):

$$\theta_\omega \leq \sigma_\omega(\bar{x}) + \sum_{i \in \bar{R}(\bar{X})} (n - \sigma_\omega(\bar{x}))x_i, \tag{A.7}$$

which is clearly valid. We refer to inequalities (A.7) as the *strengthened integer L-shaped optimality cuts*.

Proposition 14. *The submodular optimality cuts* (2.7) *dominate the combinatorial optimality cuts* (A.7).

Proof. This follows because $r_j^\omega(S) \leq n - \sigma_\omega(\bar{x})$ for any $j \in \bar{R}(S)$. □

Appendix B

COMPUTATIONS WITH BENDERS USING STRENGTHENED L-SHAPED CUTS

In our computational study in Section 2.5.2, we set $\pi_{ij} = p = 0.1, (i,j) \in A$ in the real world network. Because the influence probability p is very small, the live-arc graphs corresponding to each scenario are large-scale sparse networks. We were not able to solve even the smallest instances (with $k = 1$ and $|\Omega| = 50$) using Benders-LC after one day. To demonstrate the inefficiency of Benders-LC, we consider a sparse network under one scenario, depicted in Figure B.1 with 15 nodes and 4 directed arcs, and compare the performance of DCG-SubIneqs and Benders-LC. In other words, we let $p_1 = 1$, which leads to a deterministic problem (i.e., a unique scenario with objective θ_1). We vary the value of k from 1 to 5.

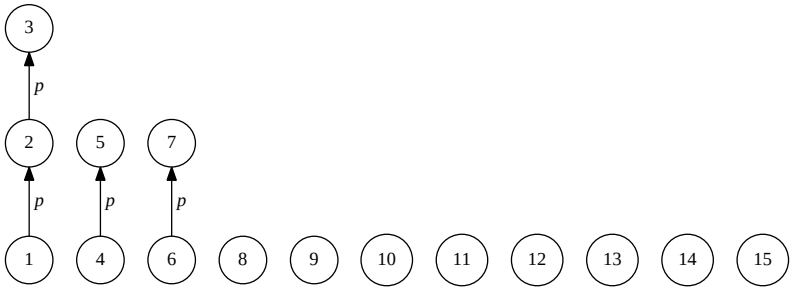

Figure B.1: Sparse Network with 15 nodes and 4 arcs with equal influence probabilities p.

The total number of user cuts added to the corresponding master problem is shown in Table B.1. We observe that compared to DCG-SubIneqs the number of user cuts added to the master problem of Benders-LC grows rapidly as the number of seed nodes k increases. Indeed, the number of user cuts for Benders-LC approached $\binom{15}{k}$, indicating that Benders-LC is effectively a pure enumeration algorithm for this problem. The strengthened integer

L-shaped optimality cuts (A.7) do not provide any useful information on the objective value when the solution is different from the one that generates the cut. In contrast, submodular inequalities are highly effective for this set of problems. To see why, consider the problem of finding $k = 1$ seed node. The master problem of both DCG-SubIneqs and Benders-LC selects $k = 1$ node arbitrarily, because they do not have any cut at the beginning. Because the sparse network is constituted of many singleton nodes (with no incoming and outgoing arcs), there is a high probability that the master problem selects one singleton at the first iteration. Suppose that the master problem chooses node 15, which was also the choice of CPLEX. DCG-SubIneqs generates the cut

$$\theta_1 \leq 1 + 3x_1 + 2x_2 + x_3 + 2x_4 + x_5 + 2x_6 + x_7 + x_8 + x_9 + x_{10} + x_{11} + x_{12} + x_{13} + x_{14},$$

and Benders-LC generates the cut

$$\theta_1 \leq 1 + \sum_{i=1}^{14} 14x_i,$$

to be added to the corresponding master problem. At the second iteration, due to the use of the stronger optimality cut, DCG-SubIneqs chooses node 1 and reaches optimality, but Benders-LC chooses one of the 14 nodes arbitrarily. Note that, in the worst case, Benders-LC traces all 15 nodes in the network (and generates 15 optimality cuts) before reaching the optimal solution. Therefore, in the large-scale network of Section 2.5.2, Benders-LC fails due to the need for a large number of iterations and computational time. In contrast, the submodular inequality guides the master problem to choose nodes with higher marginal influence.

Table B.1: Comparison of DCG-SubIneqs and Benders-LC.

Algorithm	Number of user cuts with different k				
	$k = 1$	$k = 2$	$k = 3$	$k = 4$	$k = 5$
DCG-SubIneqs	2	5	11	16	51
Benders-LC	15	106	458	1365	3003

Appendix C

THE DECOMPOSITION ALGORITHM OF LUEDTKE (2014) APPLIED TO DEP OF PSC

In this section, we give the details of the decomposition algorithm of [54] applied to DEP (3.11) of PPSC. In this algorithm, observing that for a given (x, z), the problem decomposes for each scenario, we solve a relaxed master problem (RMP) given by

$$\min \quad b^\top x \tag{C.1a}$$

$$(x, z) \in \mathcal{C}' \tag{C.1b}$$

$$\sum_{\omega \in \Omega} p_\omega z_\omega \geq 1 - \epsilon \tag{C.1c}$$

$$x \in \mathbb{B}^n, z \in \mathbb{B}^{|\Omega|}, \tag{C.1d}$$

where the set $\mathcal{C}'$ represents the set of feasibility cuts on the variables (x, z) generated until the current iteration of the algorithm. Given an incumbent solution $(\bar{x}, \bar{z})$ to RMP (C.1), for each $\omega \in \Omega$ we consider the second-stage problem

$$\min \quad 0 \tag{C.2a}$$

$$-y_i^\omega \geq -\sum_{j \in V_1} t_{ij}^\omega \bar{x}_j \qquad \forall i \in V_2 \tag{C.2b}$$

$$\sum_{i \in V_2} y_i^\omega \geq \tau \bar{z}_\omega \tag{C.2c}$$

$$y^\omega \in \mathbb{B}^m, \tag{C.2d}$$

which is a feasibility problem. We observe that the coefficient matrix of the constraints (C.2b) and (C.2c) is totally unimodular. Thus, we can relax the integrality restrictions on y_ω to obtain a second-stage linear program for each $\omega \in \Omega$

$$\min \quad 0 \tag{C.3a}$$

$$-y_i^\omega \geq -\sum_{j\in V_1} t_{ij}^\omega \bar{x}_j \qquad \forall i \in V_2 \tag{C.3b}$$

$$\sum_{i\in V_2} y_i^\omega \geq \tau \bar{z}_\omega \tag{C.3c}$$

$$-y_i^\omega \geq -1 \qquad \forall i \in V_2 \tag{C.3d}$$

$$y^\omega \in \mathbb{R}_+^m. \tag{C.3e}$$

Let π_i^1 be the dual variable associated with inequality (C.3b) for each $i \in V_2$. Let π^2 be the dual variable associated with inequality (C.3c). Let π_i^3 be the dual variable associated with inequality (C.3d) for each $i \in V_2$. Given an incumbent solution $(\bar{x}, \bar{z})$ of RMP (C.1), the dual of the second-stage subproblem is

$$\delta_\omega(\bar{x}, \bar{z}_\omega) = \max \quad -\sum_{i\in V_2} \pi_i^1 \sum_{j\in V_1} t_{ij}^\omega \bar{x}_j + \pi^2 \tau \bar{z}_\omega - \sum_{i\in V_2} \pi_i^3 \tag{C.4a}$$

$$-\pi_i^1 + \pi^2 - \pi_i^3 \leq 0 \qquad \forall i \in V_2 \tag{C.4b}$$

$$\pi^1 \in \mathbb{R}_+^m, \pi^2 \in \mathbb{R}_+, \pi^3 \in \mathbb{R}_+^m. \tag{C.4c}$$

Note that if an incumbent solution $(\bar{x}, \bar{z})$ is such that $\bar{z}_\omega = 1$ and $\delta_\omega(\bar{x}, \bar{z}_\omega) = 0$ for some $\omega \in \Omega$ (i.e., $\bar{z}_\omega = 1$ and the second-stage problem is feasible for the given $\bar{x}$), then we do not need to generate a feasibility cut for this scenario. Therefore, we only consider the case that for some $\bar{\omega} \in \Omega$, the incumbent solution $(\bar{x}, \bar{z})$ has $\bar{z}_{\bar{\omega}} = 1$, but $\delta_\omega(\bar{x}, \bar{z}_{\bar{\omega}}) = \infty$. In other words, the RMP gives a solution $\bar{z}_{\bar{\omega}} = 1$ implying that there exists a feasible solution to the second-stage subproblem for $\bar{\omega}$, however, the second-stage problem is infeasible. Then we need to add a feasibility cut to cut off this infeasible solution. Because the dual of the second-stage problem is unbounded, we have a direction of unboundedness given by the extreme ray $\bar{\pi}_{\bar{\omega}} = (\bar{\pi}^1, \bar{\pi}^2, \bar{\pi}^3)$. To obtain a feasibility cut, we solve a secondary subproblem for each $\omega \in \Omega$, given by

$$\varphi_\omega(\bar{\pi}_{\bar{\omega}}) = \sum_{i\in V_2} \bar{\pi}_i^1 \sum_{j\in V_1} t_{ij}^\omega x_j \tag{C.5a}$$

$$\sum_{j\in V_1} t_{ij}^\omega x_j \geq y_i^\omega \qquad \forall i \in V_2 \tag{C.5b}$$

$$\sum_{i \in V_2} y_i^{\omega} \geq \tau \tag{C.5c}$$

$$x \in \mathbb{B}^n, y^{\omega} \in \mathbb{B}^m. \tag{C.5d}$$

Note that the secondary subproblem is a cardinality-constrained deterministic set covering problem, which is NP-hard. After obtaining $\varphi_{\omega}(\bar{\pi}_{\bar{\omega}})$ for all $\omega \in \Omega$, we sort them in non-increasing order

$$\varphi_{\varrho 1}(\bar{\pi}_{\bar{\omega}}) \geq \varphi_{\varrho 2}(\bar{\pi}_{\bar{\omega}}) \geq \cdots \geq \varphi_{\varrho i}(\bar{\pi}_{\bar{\omega}}) \geq \cdots \geq \varphi_{\varrho |\Omega|}(\bar{\pi}_{\bar{\omega}}),$$

where ϱ is a permutation of Ω. Let $Q = \{q_1, q_2, \ldots, q_k\} \subseteq \{\varrho_1.\varrho_2, \ldots, \varrho_\ell\}$, where $\ell = \lfloor \epsilon |\Omega| \rfloor$, $\varphi_{q_i}(\bar{\pi}_{\bar{\omega}}) \geq \varphi_{q_{i+1}}(\bar{\pi}_{\bar{\omega}})$ for $i = 1, \ldots, k$, and $\varphi_{q_{k+1}}(\bar{\pi}_{\bar{\omega}}) = \varphi_{\varrho_{l+1}}(\bar{\pi}_{\bar{\omega}})$. [54] shows that

$$\sum_{i \in V_2} \bar{\pi}_i^1 \sum_{j \in V_1} t_{ij}^{\omega} x_j + \sum_{i=1}^{k} (\varphi_{q_i}(\bar{\pi}_{\bar{\omega}}) - \varphi_{q_{i+1}}(\bar{\pi}_{\bar{\omega}}))(1 - z_{q_i}) \geq \varphi_{q_1}(\bar{\pi}_{\bar{\omega}}) \tag{C.7}$$

is a valid feasibility cut that cuts off the current infeasible solution. Given an incumbent solution $(\bar{x}, \bar{z})$ of RMP (C.1), we generate inequality (C.7) for each $\bar{\omega} \in \Omega$ such that $\bar{z}_{\bar{\omega}} = 1$ and $\delta_{\omega}(\bar{x}, \bar{z}_{\bar{\omega}}) = \infty$, and add these inequalities to $\mathcal{C}'$ in RMP (C.1). This process is repeated until no such feasibility cut is needed and the incumbent solution is optimal.