SIAM J. OPTIM.
Vol. 29, No. 1, pp. 690–718

PROBABILISTIC PARTIAL SET COVERING WITH AN ORACLE FOR CHANCE CONSTRAINTS*

HAO-HSIANG WU† AND SİMGE KÜÇÜKYAVUZ‡

Abstract. We consider a class of chance-constrained combinatorial optimization problems, which we refer to as probabilistic partial set-covering (PPSC) problems. Given a prespecified risk level $\epsilon \in [0,1]$, the PPSC problem aims to find the minimum cost selection of subsets of items such that a target number of items is covered with probability at least $1-\epsilon$. We show that PPSC admits an efficient probability oracle that computes the coverage probability exactly, under certain distributions of the random variables representing the coverage relation. Using this oracle, we give a compact mixed-integer program that solves the PPSC problem for a special case. For large-scale instances for which an exact oracle-based method exhibits slow convergence, we propose a sampling-based approach that exploits the special structure of PPSC. The oracle can be used as a tool for checking and fixing the feasibility of the solution given by this approach. 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 effectively provides high-quality feasible solutions to the large-scale test instances.

Key words. chance constraints, stochastic programming, oracle, probabilistic set covering, facets, submodularity

AMS subject classifications. 90C15, 90C10

DOI. 10.1137/17M1141576

1. Introduction.

In this paper, we consider a probabilistic partial set-covering (PPSC) problem introduced in [44] 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, 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 $\tau \leq m$ and a risk level $\epsilon \in [0,1]$, the probabilistic *partial* set-covering problem aims to find the minimum-cost selection of subsets that covers 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. In this paper, we investigate methods that leverage the existence of such an efficient probability oracle.

PPSC can be modeled as a chance-constrained program (CCP). CCPs, first introduced in [8], 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 paper, we consider a class of 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 ith component of x is selected, and $x_i = 0$ otherwise. Let $\mathcal{B}(x)$ represent a random event of interest

*Received by the editors August 1, 2017; accepted for publication (in revised form) December 20, 2018; published electronically March 5, 2019.
http://www.siam.org/journals/siopt/29-1/M114157.html
Funding: The work of the authors was supported in part by the National Science Foundation grant 1907463.
†Department of Industrial and Systems Engineering, University of Washington, Seattle, WA 98195 (hhwu2@uw.edu).
‡Corresponding author. Department of Industrial Engineering and Management Sciences, Northwestern University, Evanston, IL 60208 (simge@northwestern.edu).

for a given x. Given a risk level $\epsilon \in [0,1]$, a chance-constrained program is

$$\min\{b^\top x : \mathbb{P}(\mathcal{B}(x)) \geq 1-\epsilon,\ x \in \mathcal{X} \cap \mathbb{B}^n\}, \tag{1.1}$$

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 multidimensional integrals. Second, the feasible region of CCPs is nonconvex 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 [30, 31, 6, 26, 21]. This creates an approximation of the chance constraint, which can be evaluated for the given samples. In this paper, we assume that there exists an efficient oracle, which provides an exact value of $\mathbb{P}(\mathcal{B}(x))$ for a given x. Using this oracle, a general decomposition algorithm can be applied to problem (1.1) using the true distribution instead of sampling from the true distribution. In our computational experience, such an algorithm is able to find a truly optimal solution for PPSC for small-sized 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. 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., [35, 43, 18]). 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 (nonconvexity 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 [22]. For the general case (without sampling), we consider a reformulation with exponentially many linear inequalities. This formulation can be solved 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) linear mixed-integer 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 (1.1) significantly. In particular, 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 [25], 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. A literature review of other variants of probabilistic set-covering problems and further discussions on alternative approaches are given in sections 2 and 3, respectively.

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 [29]. Van Ackooij, Frangioni, and Oliveira [36] consider such convex chance-constrained combinatorial optimization problems, where the objective function is nondifferentiable. They 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 nondifferentiable objective value. In contrast, we assume that the objective function is smooth (linear) and use an exact oracle for evaluating the *nonconvex* chance constraint. In another line of work, van Ackooij and Sagastizábal [37] 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 nonsmooth optimization (bundle) method that uses an inexact oracle to evaluate the chance constraint to find an approximate solution. In contrast, in our problem (1.1), 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 nonconvex set.

We summarize our contributions and give an outline of the paper as follows. In section 2, we introduce a class of NP-hard PPSC problems that admits an efficient probability oracle under certain probability distributions. We show that we can solve these PPSC problems by using a compact deterministic MIP under a special case. In section 3, we review a general algorithm for solving PPSC problems optimally. 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 that subsumes the known submodular inequalities. 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. We show that we can solve the sample approximation problem of PPSC by using a compact deterministic MIP under a special case. In section 4, we report the computational results with these alternative approaches. Finally, we give a summary of our results and future research directions in section 5.

2. A probabilistic partial set-covering problem with a probability oracle. We study a PPSC problem in the form (1.1). 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 [4] 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 prespecified time limit. In this case, we can form a set S_j of 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

$$\min \quad \sum_{j \in V_1} b_j x_j \tag{2.1a}$$

$$\text{s.t.} \quad \sum_{j \in V_1} t_{ij} x_j \geq h_i \qquad \forall i \in V_2, \tag{2.1b}$$

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

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 [12] proves that the set-covering problem is 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 (2.1b) 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 [7] and Saxena, Goyal, and Lejeune [33] study the uncertainty in the right-hand side of constraint (2.1b); in other words, h_i is assumed to be a binary random variable. Fischetti and Monaci [10] and Ahmed and Papageorgiou [3] study the uncertainty with the randomness in the coefficients of constraint (2.1b), i.e., t_{ij} is a binary random variable indicating if set j covers item i or not. They consider *individual* chance constraints that ensure each item is covered with a certain probability. In this paper, 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 prespecified 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

$$\min\{b^\top x : \mathbb{P}(\sigma(x) \geq \tau) \geq 1 - \epsilon,\ x \in \mathbb{B}^n\}, \tag{2.2}$$

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 [13]. 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 node $j \in V_2$ represents item j. 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 paper, we consider PPSC problems 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, Kleinberg, and Tardos [13], 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, Kleinberg, and Tardos [13], applied to bipartite graphs. Zhang et al. [44] 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. [44] can be formulated as a PPSC problem. Let $\mathcal{A}(x)$ be a probability oracle function such that $\mathcal{A}(x) = \mathbb{P}(\sigma(x) \geq \tau)$. Zhang et al. [44] describe a polynomial time algorithm to compute $\mathcal{A}(x)$ exactly for bipartite graphs under certain probability distributions for a given x. They propose a greedy heuristic to obtain a solution to the PPSC problem, 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 consider exact (respectively, approximate) algorithms to find a feasible x with exact (respectively, probabilistic) optimality guarantees. Next, we review an efficient oracle, $\mathcal{A}(x)$, for PPSC under the distributions of interest for both versions of the PPSC problem.

2.1. 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 [11, 32, 40], 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 [32] provides a formula to obtain the value of the probability mass function of the Poisson binomial distribution:

$$\mathbb{P}(\sigma(x) = j) = P(x, i) \times \mathbb{P}_i(\sigma(x) = j - 1) + (1 - P(x, i)) \times \mathbb{P}_i(\sigma(x) = j). \tag{2.3}$$

Next, we describe a dynamic program (DP) to compute $\mathbb{P}(\sigma(x) \geq \tau)$ exactly [5, 44]. 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 $0 \le j \le i$, $i \in V_2$. The DP recursion for $A(x,i,j)$ for $1 \le j \le i$, $i \in V_2$, 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, $\mathcal{A}(x) = \mathbb{P}(\sigma(x) \ge \tau) = \sum_{j=\tau}^{m} A(x,m,j)$. 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 $\mathbb{P}(\sigma(x) \ge \tau)$.

LEMMA 2.1 (see [38, Lemma 2.12]). *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 that of R_1, and the marginal distribution of $\hat{R}_2$ is the same as that of R_2. Given $\gamma \in \mathbb{R}$, $\mathbb{P}(R_1 \ge \gamma) \le \mathbb{P}(R_2 \ge \gamma)$ if and only if there exists a coupling $\hat{R}_1$ and $\hat{R}_2$ of R_1 and R_2 such that $\mathbb{P}(\hat{R}_1 \le \hat{R}_2) = 1$.*

PROPOSITION 2.2. *Given τ, $\mathbb{P}(\sigma(x) \ge \tau)$ is a monotonically increasing function in x for the PPSC 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 $P(x,i) = \sum_{j \in V_1} a_{j,i} x_j$ for the linear threshold model, and $P(x,i) = 1 - \prod_{j \in V_1} (1 - a_{j,i} x_j)$ for the independent probability coverage model. Since $P(x,i)$ is a monotone increasing function in x, $P(x',i) \le P(x'',i)$ for all $i = 1, \ldots, m$. We construct two random variables $\hat{\sigma}(x')$ and $\hat{\sigma}(x'')$ as a coupling of $\sigma(x')$ and $\sigma(x'')$. We apply the technique shown in [28, 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 $U_i \le P(x',i)$, 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 $U_i \le P(x'',i)$, 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 $P(x',i) \le P(x'',i)$, $\hat{\sigma}_i(x') \le \hat{\sigma}_i(x'')$ for all $i = 1, \ldots, m$. Then, $\hat{\sigma}(x') \le \hat{\sigma}(x'')$. From Lemma 2.1, if $\mathbb{P}(\hat{\sigma}(x') \le \hat{\sigma}(x'')) = 1$, then $\mathbb{P}(\sigma(x') \ge \tau) \le \mathbb{P}(\sigma(x'') \ge \tau)$. This completes the proof. ∎

Proposition 2.2 proves the intuitive result that if more nodes from V_1 are selected, then we have a greater chance to cover more nodes from V_2. Consequently, Proposition 2.2 allows us to use a stronger inequality in the decomposition method described in section 3. In the following subsection, we employ the DP described in this section to reformulate the problem using a compact mathematical model.

2.2. A compact MIP for PPSC with a probability oracle. Using the DP representation of the oracle $\mathcal{A}(x) = \mathbb{P}(\sigma(x) \ge \tau)$, PPSC problem (2.2) can be reformulated as a compact mathematical program,

$$\min \quad \sum_{i \in V_1} b_i x_i \tag{2.4a}$$

$$\text{(2.4b)} \qquad \text{s.t.} \quad \bar{A}_{0,0} = 1,$$

$$\text{(2.4c)} \qquad \bar{A}_{i,j} = \bar{A}_{i-1,j}(1 - P(x,i)), \quad i = 1,\dots,m,\ j = 0,$$

$$\text{(2.4d)} \qquad \bar{A}_{i,j} = \bar{A}_{i-1,j}(1 - P(x,i)) + \bar{A}_{i-1,j-1}P(x,i), \quad i = 1,\dots,m,\ 0 < j < i,$$

$$\text{(2.4e)} \qquad \bar{A}_{i,j} = \bar{A}_{i-1,j-1}P(x,i), \quad i = 1,\dots,m,\ j = i,$$

$$\text{(2.4f)} \qquad \sum_{j=\tau}^{m} \bar{A}_{m,j} \geq 1 - \epsilon,$$

$$\text{(2.4g)} \qquad x \in \mathbb{B}^n,$$

$$\text{(2.4h)} \qquad \bar{A}_{i,j} \in \mathbb{R}_+, \quad 0 \leq j \leq i \leq m,$$

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 (2.4b) is the boundary condition of the DP, constraints (2.4c)–(2.4e) are the DP recursive functions, and constraint (2.4f) is the goal function. Note that $P(x,i)$ is a function of the decision vector x. Hence, formulation (2.4) is a *nonlinear* mixed-integer program due to the constraints (2.4c)–(2.4e). Depending on the complexity of the function $P(x,i)$, this formulation may be difficult to solve. However, for a special case of the PPSC problem, 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 (2.4c)–(2.4e) is a bilinear term, which can be linearized [23, 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:

$$\text{(2.5a)} \qquad \min \quad \sum_{i \in V_1} b_i x_i$$

$$\text{(2.5b)} \qquad \text{s.t.} \quad \text{(2.4b), (2.4f)–(2.4h)},$$

$$\text{(2.5c)} \qquad \bar{A}_{i,j} = \bar{A}_{i-1,j} - \sum_{u \in V_1} a_{u,i}\gamma_{u,i-1,j}, \quad i = 1,\dots,m,\ j = 0,$$

$$\text{(2.5d)} \qquad \bar{A}_{i,j} = \bar{A}_{i-1,j} - \sum_{u \in V_1} a_{u,i}\gamma_{u,i-1,j} + \sum_{u \in V_1} a_{u,i}\gamma_{u,i-1,j-1}, \quad i = 1,\dots,m,\ 0 < j < i,$$

$$\text{(2.5e)} \qquad \bar{A}_{i,j} = \sum_{u \in V_1} a_{u,i}\gamma_{u,i-1,j-1}, \quad i = 1,\dots,m,\ j = i,$$

$$\text{(2.5f)} \qquad \gamma_{u,i,j} \leq x_u, \quad i = 0,\dots,m,\ j = 0,\dots,i,\ u \in V_1,$$

$$\text{(2.5g)} \qquad \gamma_{u,i,j} \leq \bar{A}_{i,j}, \quad i = 0,\dots,m,\ j = 0,\dots,i,\ u \in V_1,$$

$$\text{(2.5h)} \qquad \gamma_{u,i,j} \geq \bar{A}_{i,j} - (1 - x_u), \quad i = 0,\dots,m,\ j = 0,\dots,i,\ u \in V_1,$$

$$\text{(2.5i)} \qquad \gamma_{u,i,j} \in \mathbb{R}_+, \quad i = 0,\dots,m,\ j = 0,\dots,i,\ u \in V_1.$$

The DP recursion is represented in constraints (2.5c)–(2.5e). Constraints (2.5f)–(2.5i) 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 (2.5). We close this subsection by noting that, for the case of the independent probability coverage model, the mixed-integer nonlinear program (2.4) 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 applications of the McCormick linearization, the resulting formulations are large scale and suffer from weak LP relaxations. Therefore, we do not pursue such formulations for the independent probability coverage model in our computational study.

3. Decomposition approaches for PPSC with a probability oracle. In this section, we review exact and approximate decomposition methods for PPSC that admits an efficient probability oracle.

3.1. An exact decomposition method. Using the probability oracle we have just described, we first review a general algorithm to solve formulation (2.2) along the lines of [15] (see also [34] and the online supplement of [35], for an application to binary packing problems). Consider the generic relaxed master problem (RMP) of formulation (2.2) as

$$\min\{b^\top x : x \in \mathcal{C} \cap \mathcal{X} \cap \mathbb{B}^n\}, \tag{3.1}$$

where $\mathcal{C}$ is a set of feasibility cuts added until the current iteration. The algorithm works by solving a relaxed problem and cutting off infeasible solutions iteratively until we find an optimal solution. We describe a delayed constraint generation approach with the probability oracle in Algorithm 1, starting with a subset of feasibility cuts in $\mathcal{C}$ (which could be empty). At each iteration (lines 2–7), solving RMP (3.1) 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 8); 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 7) to cut off $\bar{x}$ in further iterations. We describe this subroutine in Algorithm 1 and specify the corresponding cuts in the following paragraphs.

Algorithm 1. A general algorithm for PPSC.

```
1 Start with an initial set of feasibility cuts in C (could be empty);
2 while True do
3 |   solve master problem (3.1), and obtain an incumbent solution x̄;
4 |   if A(x̄) ≥ 1 − ϵ then
5 |   |   break;
6 |   else
7 |   |   call FeasibilityCut(x̄, κ, C);
8 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

$$\sum_{i\in J_1}(1-x_i) + \sum_{j\in J_0} x_j \geq 1; \tag{3.2}$$

this ensures that if $\bar{x}$ is infeasible, then at least one component in $\bar{x}$ must be changed [15]. In the next proposition, we observe that if $\mathcal{A}(x)$ is monotonically increasing in x for problem (1.1), then a stronger inequality is valid for formulation (2.2). Throughout, we let $\mathbf{e}_j$ be a unit vector of dimension n whose jth component is 1.

PROPOSITION 3.1. *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*

$$\sum_{j\in J_0} x_j \geq \kappa(J_0) \tag{3.3}$$

is valid for formulation (2.2).

(ii) *Inequality* (3.3) *is stronger than inequality* (3.2) *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 (2.2). 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 the light of Proposition 3.1, we specify the subroutine FeasibilityCut$(\bar{x}, \kappa, \mathcal{C})$ in Algorithm 2 for the case that $\mathcal{A}(\bar{x})$ is monotone. If $\mathcal{A}(\bar{x})$ is not monotone, then inequality (3.3) should be replaced with inequality (3.2) in Algorithm 2. In this subroutine, we seek inequalities (3.3) with $\kappa(J_0) \leq \kappa$, given the input parameter κ. If $\kappa = 2$, then we check whether 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 (3.3). Note that values of κ higher than 2 will require more computational effort, so we only consider $\kappa = 2$ in Algorithm 2.

Algorithm 2. Subroutine FeasibilityCut$(\bar{x}, \kappa, \mathcal{C})$.

```
for j ∈ J_0 do
    if A(x̄ + e_j) ≥ 1 − ε then
        add inequality (3.3) with κ(J_0) = 1 to C;
        BoundIncrease = 0;
        break;
if κ = 2 then
    add inequality (3.3) with κ(J_0) = 2 to C.
```

Algorithm 1 can be used to solve formulation (2.2) exactly. To update $\kappa(J_0)$ in Algorithm 2 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 $\mathcal{O}(m)$.

When the number of decision variables is large, Algorithm 1 exhibits slow convergence, even if there exists an efficient probability oracle. In this case, Algorithm 1 may check a large (worst-case exponential) number of incumbent solutions $\bar{x}$ to obtain the optimal solution. This motivates us to consider a sampling-based approach to find approximate solutions to the PPSC problem. This 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.

3.2. 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 nonconvex 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, Ahmed, and Nemhauser [22], Küçükyavuz [14], Abdi and Fukasawa [1] Zhao, Huang, and Zeng [45], and Liu, Kılınç-Karzan, and Küçükyavuz[17] introduce strong valid inequalities for the deterministic equivalent formulation of linear chance constraints under right-hand side uncertainty. Ruszczyński [31], Beraldi and Bruni [6], Lejeune [16], Luedtke [20], and Liu, Küçükyavuz, and Luedtke [19] 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, Luedtke, and Küçükyavuz [35] 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 the PPSC problem. First, we describe how we sample from the true distribution to obtain a set of scenarios (sample paths) Ω for PPSC (see [13, 42] 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 whether 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, Kleinberg, and Tardos [13] 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 $t^\omega_{ij} = 1$ if arc $(i, j) \in E_\omega$ for $\omega \in \Omega$, and $t^\omega_{ij} = 0$ otherwise.

Given a set of scenarios Ω and a target level τ, we can reformulate the submodular formulation (2.2) 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 the PPSC problem is the following:

$$\text{(3.4a)} \quad \min \quad \sum_{j \in V_1} b_j x_j$$

$$\text{(3.4b)} \quad \text{s.t.} \quad \sum_{j \in V_1} t_{ij}^\omega x_j \geq y_i^\omega \qquad \forall i \in V_2,\ \forall \omega \in \Omega,$$

$$\text{(3.4c)} \quad \sum_{i \in V_2} y_i^\omega \geq \tau z_\omega \qquad \forall \omega \in \Omega,$$

$$\text{(3.4d)} \quad \sum_{\omega \in \Omega} p_\omega z_\omega \geq 1 - \epsilon,$$

$$\text{(3.4e)} \quad x \in \mathbb{B}^n,\ z \in \mathbb{B}^{|\Omega|},\ 0 \leq y_i^\omega \leq 1 \quad \forall i \in V_2,\ \forall \omega \in \Omega,$$

where constraints (3.4b) 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 (3.4c) ensure that if $z_\omega = 1$, then $\sum_{i \in V_2} y_i^\omega \geq \tau$ for all $\omega \in \Omega$. Constraint (3.4d) ensures that the probability that τ items are covered in V_2 is at least $1-\epsilon$. Note that, for a given integer vector (x, z), the coefficient matrix of the constraints (3.4b) and (3.4c) is totally unimodular. Therefore, we can relax the integrality of the y variables in formulation (3.4).

Formulation (3.4) is a very large-scale MIP that continues to challenge state-of-the-art optimization solvers. Instead, delayed constraint generation methods akin to the Benders decomposition method [20, 19] are known to be computationally more effective for such problems. Because the second-stage problem is concerned only with feasibility, the decomposition algorithm proposed in [20] is applicable to this formulation (Liu, Küçükyavuz, and Luedtke [19] also consider the second-stage objective). However, we observe that the deterministic set-covering problem, which is known to be NP-hard, can be reduced to one of the subproblems required for this algorithm. 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 paper, we propose an alternative approach and use the submodularity property of PPSC to solve formulation (3.4), 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 [39, 13]. Given a set of scenarios Ω and target level τ, we formulate PPSC as

$$\text{(3.5a)} \quad \min \quad \sum_{i \in V_1} b_i x_i$$

$$\text{(3.5b)} \quad \text{s.t.} \quad \sigma_\omega(x) \geq \tau z_\omega, \qquad \omega \in \Omega,$$

$$\text{(3.5c)} \quad \sum_{\omega \in \Omega} p_\omega z_\omega \geq 1 - \epsilon,$$

$$\text{(3.5d)} \quad x \in \mathbb{B}^n,\ z \in \mathbb{B}^{|\Omega|},$$

where $z_\omega = 1$ implies that, for a given x, $\sigma_\omega(x) \geq \tau$ is enforced. Constraint (3.5c) ensures that the probability that $\sigma_\omega(x) \geq \tau$ is at least $1-\epsilon$. Constraint (3.5b) involves

a submodular function. To reformulate it using linear inequalities, we introduce an additional variable θ_ω that represents the number of covered nodes in V_2 under each 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

$$\mathcal{S}_\omega = \left\{(\theta_\omega, x) \in \mathbb{R} \times \{0,1\}^n : \theta_\omega \leq \sigma_\omega(S) + \sum_{j \in V_1 \setminus S} \rho_j^\omega(S) x_j \ \forall S \subseteq V_1\right\},$$

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 [24] 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 the PPSC problem. The corresponding relaxed RMP is defined as

$$\min \quad \sum_{i \in V_1} b_i x_i \tag{3.6a}$$

$$\text{s.t.} \quad (\theta_\omega, x) \in \bar{\mathcal{C}}, \tag{3.6b}$$

$$\theta_\omega \geq \tau z_\omega, \qquad \omega \in \Omega, \tag{3.6c}$$

$$\sum_{\omega \in \Omega} p_\omega z_\omega \geq 1 - \epsilon, \tag{3.6d}$$

$$x \in \mathbb{B}^n,\ z \in \mathbb{B}^{|\Omega|},\ \theta \in \mathbb{R}_+^{|\Omega|}, \tag{3.6e}$$

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 an incumbent solution, $\bar{x}$, of RMP (3.6), and its corresponding support $\bar{X} = \{i \in V_1 : \bar{x}_i = 1\}$, a submodular feasibility cut [24, 25] is

$$\theta_\omega \leq \sigma_\omega(\bar{X}) + \sum_{j \in V_1 \setminus \bar{X}} \rho_j^\omega(\bar{X}) x_j. \tag{3.7}$$

At each iteration of the algorithm, we solve RMP (3.6) to obtain an incumbent solution $(\bar{x}, \bar{\theta}, \bar{z})$, which is used to generate the submodular cuts (3.7), if necessary. In a related study, Wu and Küçükyavuz [42] apply inequality (3.7) 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 (nonbipartite) network. Wu and Küçükyavuz [42] give conditions under which inequalities (3.7) are facet defining for $\mathcal{S}_\omega$. We extend the work of [42], and propose a new class of valid inequalities for the bipartite case. Before we give our proposed inequality, we provide some useful definitions.

DEFINITION 3.2. *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 3.3. *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 given that $C_2^i \cap C_2^j = \emptyset$ for all $i = 1, \ldots, c$ and $j = 1, \ldots, c$ with $i \neq j$, the inequality*

$$\theta_\omega \leq \sum_{k=1}^{c} n_\omega(C_1^k)\left(1 - \sum_{j \in C_1^k} x_j\right) + \sum_{k \in D} \eta_\omega^k (1 - x_k) + \sum_{j \in V_1} \sigma_\omega(\{j\}) x_j \tag{3.8}$$

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 $i \in \hat{X}$, and 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

$$\begin{aligned}
\hat{\theta}_\omega &\leq \sigma_\omega(\hat{X}) \\
&= \sum_{j \in \hat{X}} \sigma_\omega(\{j\}) - \sum_{v \in V_2} \sum_{j \in \hat{X} \setminus \{r(v)\}} |v \cap \mathcal{U}_\omega(\{j, r(v)\}, \emptyset)| && (3.9) \\
&= \sum_{j \in V_1} \sigma_\omega(\{j\}) \hat{x}_j - \sum_{v \in V_2} \sum_{j \in V_1 \setminus \{r(v)\}} |v \cap \mathcal{U}_\omega(\{j, r(v)\}, \emptyset)| \hat{x}_j && (3.10) \\
&\leq \sum_{j \in V_1} \sigma_\omega(\{j\}) \hat{x}_j - \sum_{k \in C''} \sum_{v \in C_2^k} \sum_{j \in V_1 \setminus \{r(v)\}} |v \cap \mathcal{U}_\omega(\{j, r(v)\}, \emptyset)| \hat{x}_j && (3.11) \\
&\leq \sum_{j \in V_1} \sigma_\omega(\{j\}) \hat{x}_j - \sum_{k \in C''} \sum_{v \in C_2^k} \sum_{j \in C_1^k \cap \{V_1 \setminus \{r(v)\}\}} |v \cap \mathcal{U}_\omega(\{j, r(v)\}, \emptyset)| \hat{x}_j && (3.12) \\
&\leq \sum_{j \in V_1} \sigma_\omega(\{j\}) \hat{x}_j - \sum_{k \in C''} \sum_{v \in C_2^k} \sum_{j \in C_1^k \cap \{V_1 \setminus \{r_k\}\}} |v \cap \mathcal{U}_\omega(\{j, r_k\}, \emptyset)| \hat{x}_j && (3.13) \\
&= \sum_{j \in V_1} \sigma_\omega(\{j\}) \hat{x}_j - \sum_{k \in C''} \sum_{j \in C_1^k \cap \{V_1 \setminus \{r_k\}\}} |C_2^k \cap \mathcal{U}_\omega(\{j, r_k\}, \emptyset)| \hat{x}_j && (3.14)
\end{aligned}$$

$$\leq \sum_{j\in V_1} \sigma_\omega(\{j\})\hat{x}_j - \sum_{k\in C''} \sum_{j\in C_1^k\cap\{V_1\setminus\{r_k\}\}} |C_2^k \cap \mathcal{U}_\omega(\{j,r_k\}, V_1\setminus C_1^k)|\hat{x}_j \tag{3.15}$$

$$= \sum_{j\in V_1} \sigma_\omega(\{j\})\hat{x}_j - \sum_{k\in C''} \sum_{j\in C_1^k\cap\{V_1\setminus\{r_k\}\}} n_\omega(C_1^k)\hat{x}_j \tag{3.16}$$

$$= \sum_{j\in V_1} \sigma_\omega(\{j\})\hat{x}_j - \sum_{k\in C''} \sum_{j\in C_1^k\cap\{V_1\setminus\{r_k\}\}} n_\omega(C_1^k)\hat{x}_j + \sum_{k\in C''} n_\omega(C_1^k)(1-\hat{x}_{r_k}) \tag{3.17}$$

$$= \sum_{j\in V_1} \sigma_\omega(\{j\})\hat{x}_j - \sum_{k\in C''} \sum_{j\in C_1^k\cap V_1} n_\omega(C_1^k)\hat{x}_j + \sum_{k\in C''} n_\omega(C_1^k) \tag{3.18}$$

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

$$+ \sum_{k\in C'} n_\omega(C_1^k)\Big(1 - \sum_{j\in C_1^k} \hat{x}_j\Big) \tag{3.19}$$

$$\leq \sum_{j\in V_1} \sigma_\omega(\{j\}) + \sum_{k=1}^{c} n_\omega(C_1^k)\Big(1 - \sum_{j\in C_1^k} \hat{x}_j\Big) + \sum_{k\in D} \eta_\omega^k(1-\hat{x}_k). \tag{3.20}$$

Equality (3.9) follows from the definition of $\sigma_\omega(\hat{X})$ for a given $\hat{X}$. Equality (3.10) 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 (3.11) 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 (3.12) 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 (3.13) 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 (3.14) by reorganizing the terms in inequality (3.13). Inequality (3.15) 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 (3.16) 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 (3.17) follows from $\hat{x}_{r_k} = 1$. We obtain equality (3.18) by reorganizing the terms in inequality (3.17). Inequality (3.19) follows from the assumption that $\sum_{j\in C_1^k} x_j \leq 1$ for each $k \in C'$. Finally, inequality (3.20) follows from $\eta_\omega^k \geq 0$, $x_k \in \{0, 1\}$, and $k \in D$. This completes the proof. ∎

Example 3.1. 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 (3.8) given in Table 3.1. The corresponding inequality (3.8) is

$$\theta_\omega \leq 5 + 0x_1 + 0x_2 + 0x_3 + x_4, \tag{3.21}$$

which is equivalent to a submodular inequality (3.7) with $\bar{X} = \{1, 2, 3\}$.

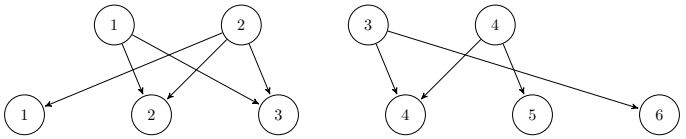

FIG. 3.1. *An example of a bipartite graph with four sets and six items.*

TABLE 3.1
The choice of parameters for inequality (3.8) *with* $D \neq \emptyset$.

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

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 (3.21). In addition, this inequality cannot be generated as a submodular inequality (3.7) for any selection of $\bar{X}$. We formalize this observation next.

PROPOSITION 3.4. *Inequalities* (3.8) *subsume the submodular inequalities* (3.7).

Proof. We show that a submodular inequality (3.7) for a given $\bar{X} \subseteq V_1$ can be represented as a corresponding inequality (3.8). 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 (3.8) 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 $j \in V_1 \setminus \bar{X}$, the coefficient of x_j in inequality (3.8) 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 (3.8) 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 (3.8) 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 (3.8). Example 3.1 shows that there are inequalities (3.8) that cannot be written as submodular inequalities (3.7). This completes the proof. ∎

Next, we provide a necessary condition for inequality (3.8) to be facet defining.

PROPOSITION 3.5. *Inequality* (3.8) *is facet defining for* $\text{conv}(\mathcal{S}_\omega)$ *only if* $D = \emptyset$.

Proof. We show that inequality (3.8) with $D = \emptyset$ given by

$$\theta_\omega \leq \sum_{k=1}^{c} n_\omega(C_1^k)\left(1 - \sum_{j \in C_1^k} x_j\right) + \sum_{j \in V_1} \sigma_\omega(\{j\}) x_j \tag{3.22}$$

dominates inequality (3.8) 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 (3.8) with $D \neq \emptyset$ has the same terms as inequality (3.8) with $D = \emptyset$, as well as the additional term $\sum_{k \in D} \eta_\omega^k (1 - x_k) \geq 0$, because $x_k \in \{0, 1\}$ and $\eta_\omega^k \geq 0$. Hence, we need to have $D = \emptyset$ for inequality (3.8) to be facet defining for $\text{conv}(\mathcal{S}_\omega)$. ∎

Note that we allow $D \neq \emptyset$ in the definition of inequality (3.8) to be able to show that inequality (3.8) subsumes submodular inequality (3.7). However, we see from the necessary condition in Proposition 3.5 that it suffices to consider inequalities (3.8) with $D = \emptyset$. Next we give some sufficient conditions for inequality (3.22) to be facet defining for $\text{conv}(\mathcal{S}_\omega)$.

PROPOSITION 3.6. *Inequality* (3.22) *is facet defining for* $\text{conv}(\mathcal{S}_\omega)$ *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* $\mathcal{U}_\omega(\{i, j\}, \emptyset) = \mathcal{U}_\omega(\{i, j\}, V_1 \setminus C_1^k) \subseteq C_2^k$, $\mathcal{U}_\omega(\{i, r\}, \emptyset) = \mathcal{U}_\omega(\{j, r\}, \emptyset) = \emptyset$ *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 (3.22) 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 (3.22) next.

Consider a point

$$(\theta_\omega, x)^0 = \left(\sum_{k=1}^c \sigma_\omega(\{f_1^k\}) + \sum_{k=1}^c \sigma_\omega(\{f_2^k\}) - \sum_{k=1}^c n_\omega(C_1^k), \beta_0 \right),$$

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

$$\mathcal{U}_\omega(\{f_1^k, r\}, \emptyset) = \mathcal{U}_\omega(\{f_2^k, r\}, \emptyset) = \emptyset \ \forall r \in V_1 \setminus C_1^k,$$

we have

$$\begin{aligned} \sigma_\omega(\bar{S}(\beta_0)) &= \sum_{k=1}^c (\sigma_\omega(\{f_1^k\}) + \sigma_\omega(\{f_2^k\}) - |\mathcal{U}_\omega(\{f_1^k, f_2^k\}, \emptyset)|) \\ &= \sum_{k=1}^c (n_\omega(C_1^k) + \sigma_\omega(\{f_1^k\}) - n_\omega(C_1^k) + \sigma_\omega(\{f_2^k\}) - n_\omega(C_1^k)), \end{aligned}$$

and hence (θ_ω, β_0) is on the face defined by inequality (3.22).

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

$$(\theta_\omega, x)^i = \left(\sum_{k=1}^c \sigma_\omega(\{f_1^k\}) + \sum_{k=1}^c \sigma_\omega(\{f_2^k\}) - \sum_{k=1}^c n_\omega(C_1^k) + \sigma_\omega(\{i\}), \beta_i^{\bar{L}} \right)$$

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

$$\begin{aligned}\sigma_\omega(\bar{S}(\beta_i^{\bar{L}})) &= \sigma_\omega(\bar{S}(\beta_0)) + \sigma_\omega(\{i\}) \\ &= \sum_{k=1}^{c} \sigma_\omega(\{f_1^k\}) + \sum_{k=1}^{c} \sigma_\omega(\{f_2^k\}) - \sum_{k=1}^{c} |\mathcal{U}_\omega(\{f_1^k, f_2^k\}, \emptyset)| + \sigma_\omega(\{i\}) \\ &= \sum_{k=1}^{c} (n_\omega(C_1^k) + \sigma_\omega(\{f_1^k\}) - n_\omega(C_1^k) + \sigma_\omega(\{f_2^k\}) - n_\omega(C_1^k)) + \sigma_\omega(\{i\}),\end{aligned}$$

and hence $(\theta_\omega^i, \beta_i^{\bar{L}})$ for all $i \in \bar{L}$ are on the face defined by inequality (3.22).

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

$$(\theta_\omega, x)^{ik} = \left(\sum_{j \in \bar{C} \setminus \{k\}} (\sigma_\omega(\{f_1^k\}) + \sigma_\omega(\{f_2^k\}) - n_\omega(C_1^j)) + n_\omega(C_1^k) + \sigma_\omega(\{i\}) - n_\omega(C_1^k), \beta_i^k \right)$$

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

$$\begin{aligned}\sigma_\omega(\bar{S}(\beta_i^k)) &= \sigma_\omega(\bar{S}(\beta_0)) - \sigma_\omega(\{f_1^k\}) - \sigma_\omega(\{f_2^k\}) + n_\omega(C_1^k) + \sigma_\omega(\{i\}) - n_\omega(C_1^k) \\ &= \sum_{j \in \bar{C} \setminus \{k\}} (n_\omega(C_1^j) + \sigma_\omega(\{f_1^j\}) - n_\omega(C_1^j) + \sigma_\omega(\{f_2^j\}) - n_\omega(C_1^j)) \\ &\quad + n_\omega(C_1^k) + \sigma_\omega(\{i\}) - n_\omega(C_1^k),\end{aligned}$$

and hence $(\theta_\omega, \beta_i^k)^{ik}$ for all $i \in C_1^k$, $k = 1, \ldots, c$, are on the face defined by inequality (3.22). These $1 + |\bar{L}| + \sum_{k=1}^{c} |C_1^k| = n + 1$ points are affinely independent. ∎

Algorithm 3 describes a sampling-based method to solve the PPSC problem by using inequalities (3.7) or (3.8) as feasibility cuts. The proposed algorithm includes the Benders phase (lines 2–9) and the oracle phase (lines 10–12). Algorithm 3 starts with a given set of feasibility cuts, $\bar{\mathcal{C}}$. In the Benders phase, master problem (3.6) 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 (3.7) or (3.8) for ω under the condition in line 8. 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 10–12). We use inequality (3.3) with $\kappa(J_0) \leq \kappa$ as the feasibility cut to cut off infeasible $\bar{x}$ until master problem (3.6) provides a truly feasible solution to the original (nonsampled) 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 (3.7) as in [42], if infeasible. Next we describe how to generate a violated new valid inequality (3.22) with $D = \emptyset$ (due to the necessary facet condition in Proposition 3.5), 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

Algorithm 3. Sampling-based delayed constraint generation algorithm with a probability oracle for PPSC.

```
Input: κ ∈ {1, 2}. Start with C̄ = {0 ≤ θ_ω ≤ m, ω ∈ Ω};
while True do
    solve master problem (3.6) and obtain an incumbent solution (x̄, θ̄, z̄);
    if Σ_{ω∈Ω} {p_ω : σ_ω(x̄) ≥ τ} ≥ 1 − ε then
        break;
    else
        for ω ∈ Ω do
            if τ > σ_ω(x̄) and θ_ω > σ_ω(x̄) then
                add a feasibility cut (3.7) or (3.8) to C̄ in master problem (3.6);
while A(x̄) < 1 − ε do
    call FeasibilityCut(x̄, κ, C̄);
    solve master problem (3.6) and obtain an incumbent solution x̄;
output x̄ as an optimal solution.
```

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 such 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 (3.22) can be generated in polynomial time.

Finally, for the PPSC problem with the linear threshold model, given a set of sampled scenarios, we observe that a polynomial number of submodular inequalities (3.7) or inequalities (3.8) is sufficient to reach an optimal solution of the master problem (3.6). In the following propositions, we summarize this result and its consequence of providing a compact MIP to solve the PPSC problem with the linear threshold model.

PROPOSITION 3.7. *For the PPSC problem with the linear threshold model, adding the submodular inequalities* (3.7) *with* $\bar{X} = \emptyset$*, which are equivalent to* (3.22) *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* (3.6).

Proof. In the live-arc graph scenario generation method proposed by Kempe, Kleinberg, and Tardos [13] 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 (3.22), 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 (3.7) with $\bar{X} = \emptyset$ are equivalent to the new inequalities (3.22) for any choice of parameters, and they are given by

$$\theta_\omega \leq \sum_{i \in V_1} \sigma_\omega(\{i\}) x_i. \tag{3.23}$$

Any submodular inequality (3.7) 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 (3.23). This completes the proof. ∎

PROPOSITION 3.8. *For PPSC under the linear threshold model, given a set of scenarios Ω, the master problem formulation* (3.6) *and the deterministic equivalent formulation* (3.4) *can be reduced to the following formulation in (x, z)-space:*

$$\min \quad \sum_{i \in V_1} b_i x_i \tag{3.24a}$$

$$s.t. \quad \sum_{i \in V_1} \sigma_\omega(\{i\}) x_i \geq \tau z_\omega, \qquad \omega \in \Omega, \tag{3.24b}$$

$$\sum_{\omega \in \Omega} p_\omega z_\omega \geq 1 - \epsilon, \tag{3.24c}$$

$$x \in \mathbb{B}^n, \ z \in \mathbb{B}^{|\Omega|}. \tag{3.24d}$$

Proof. In Proposition 3.7, we show that adding inequalities (3.23) to the master problem (3.6) 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 (3.23) and (3.6c), leading to inequalities (3.24b) and the formulation (3.24).

Next we show that the deterministic equivalent formulation (3.4) can be reduced to formulation (3.24) 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 [13]. In formulation (3.4), by summing the constraints (3.4b) 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

$$\sum_{j \in V_1} \sigma_\omega(\{j\}) x_j \geq \sum_{i \in V_2} y_i^\omega \ \forall \omega \in \Omega. \tag{3.25}$$

Now we can project out the y variables using the constraints (3.25) and (3.4c), and obtain the constraints (3.24b) and the formulation (3.24). This completes the proof. ∎

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 an 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. The CPLEX presolve process is turned off for the traditional branch-and-cut method for solving the decomposition algorithms. The relative MIP gap tolerance of CPLEX is set to the default value, so a feasible solution that has an optimality gap of $10^{-4}\%$ is considered optimal. The time limit is set to one hour.

Our dataset is motivated by the human sexual contact network (human interaction network) introduced in [9, 27]. 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 are split into two parts. In the first part, we test both the exact delayed constraint generation algorithm given in Algorithm 1, and the sampling-based approach described in Algorithm 3 in order to solve the PPSC problem under the independent probability coverage model. In the second part, we compare the exact delayed constraint generation algorithm and the deterministic equivalent MIP formulation (2.5) to solve the PPSC problem under the linear threshold model. In addition, the compact MIP model (3.24) described in Proposition 3.8 is applied to the PPSC problem under the linear threshold model. Here we summarize our main observations from a more extensive computational study. We refer the reader to [41, Chapter 3] for a more detailed analysis and additional experiments.

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 (2.4) 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 smaller expected number of items than each node $i \in V_1^1$, we set a lower cost range for $j \in V_1^2$ than for $i \in V_1^1$, 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 the PPSC problem exactly under the independent probability coverage model by using Algorithm 1, which is referred to as "Oracle." To show the effect of the choice of $\kappa(J_0)$ in inequality (3.3) 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 4.1 provides the comparison between the two methods: the "Cuts" column shows the total number of user cuts added to the master problem and the "Time" column shows the solution time in seconds. Table 4.1 shows that using the stronger no-good cuts (i.e., Oracle ($\kappa = 2$)) drastically reduces the solution time and the number of cuts required compared to the traditional no-good cuts with the additional strengthening due to the monotone property, 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 6 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 (3.3) by using a larger right-hand side ($\kappa(J_0) = 2$ versus $\kappa(J_0) = 1$). We observe that the instances with nonunit 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 nonunit 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 4.1. 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.

TABLE 4.1
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 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, which is described in section 3.2. We demonstrate the usage of the probability oracle for checking and fixing the feasibility of the solution given by the sampling-based approach for instances with $|V| = 120$. For the instances with $\bar{b} = 1$, we generate $|\Omega| = \{100, 500, 1000\}$ equiprobable scenarios. However, for $\bar{b} = 100$, we generate fewer equiprobable scenarios ($|\Omega| = \{100, 250, 500\}$), because the instances with $\bar{b} = 100$ are harder to solve than the instances with $\bar{b} = 1$. For each combination of $(\bar{b}, \epsilon, |\Omega|)$, we create three replications of the scenario set and report the average statistics. We consider the sampling-based delayed constraint generation method (Algorithm 3), which is referred to as delayed cut generation (DCG) in this subsection. Recall that Algorithm 3 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 (3.7) (referred to as DCG-Sub) and new valid inequality (3.22) (referred to as DCG-NV), to RMP (3.6). 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 2.1. We use Algorithm 2 with $\kappa = 2$ in these experiments to add feasibility cuts (3.3) to RMP (3.6). We also consider the deterministic equivalent problem (3.4) using the linear representation of the chance constraint. In the case of DEP (3.4),

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 2.1 as an oracle. If the current solution is not feasible with respect to the true distribution, then we add inequality (3.3) 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.

In Table 4.2, we compare the performances of DCG-NV, DCG-Sub, and DEP (3.4) and demonstrate the utility of the oracle in the sampling-based delayed constraint generation algorithm. The "Master" columns report the statistics pertaining to the Benders phase of DCG, and the "DEP" columns show the deterministic equivalent problem (3.4). The "Oracle" columns report the statistics pertaining to the oracle phase of DCG and DEP (3.4). Note that we set a one-hour time limit for both Master and DEP, and for the oracle phase. In the "Master," "DEP" and "Oracle" columns, "Time(u)" denotes the solution time in seconds ("(u)" denotes the number of unsolved instances out of the three instances tested for the corresponding setting). In the "Master," column "Cuts" denotes number of inequalities (3.7) and (3.22) added to RMP (3.6) for DCG-Sub and DCG-NV, respectively. In the "Oracle," column, "Cuts" shows the number of inequalities (3.3) added to RMP (3.6) in the oracle phase. The "Nodes" column shows the number of branch-and-bound nodes traced in the Benders phase. For the instances that do not solve within the time limit, the "Gap" column 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. The "Inf" column shows the number of instances, among those for which an optimal solution was found in the Benders phase, that provide 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 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 phase are deemed feasible by the oracle phase (indicated by Inf = 0), then we put a dash (—) in 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 in parentheses the number of unsolved instances "(u)" out of the total number of instances tested in the oracle phase (given by "Inf").

Table 4.2 shows that the solution time increases as ϵ and $|\Omega|$ increase. We also note that the problems with nonunit 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 when DCG can provide an optimal solution within the time limit, DCG-NV is faster than DCG-Sub in most cases. In addition, the "Cuts" and "Nodes" columns 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 (3.22), which we prove to be a stronger inequality than inequality (3.7) (Proposition 3.5), improves the computational performance of DCG.

Next, we demonstrate the usage of the oracle for checking and fixing the feasibility of the solution given by the sampling-based approach. In Oracle, Inf = 0 denotes that the solution provided at the end of the Benders phase is feasible. The instances that

TABLE 4.2
Networks with $|V| = 120$ for PPSC with the independent probability coverage model sampling.

			DCG-NV					DCG-Sub						DEP (3.4)					
			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
1	0.025	100	≤ 1	695	604	0	2	7	101	≤ 1	871	885	0	2	7	114	7	49	0	2	20	3
		500	180	4069	29550	0	0	—	—	255	6187	31508	0	0	—	—	1714	5332	0	0	—	—
		1000	859	6083	84757	0	0	—	—	1740	9969	126428	0	0	—	—	(3)	1307	30.7	—	—	—
		100	3	836	1192	0	3	13	114	3	849	1496	0	3	21	134	12	119	0	3	30	10
		500	866	7084	100719	0	0	—	—	1206	9151	114554	0	0	—	—	(3)	9905	23.4	—	—	—
		1000	(3)	11827	128201	39.59	—	—	—	(3)	16492	83493	42.58	—	—	—	(3)	832	33.33	—	—	—
	0.05	100	17	954	7141	0	3	18	1	18	1282	8112	0	2	13	6	27	425	0	1	37	9
		500	2426	4138	246194	0	2	1674	14	2461(1)	5139	354775	20	1	895	1	1186(1)	1534	23.93	1	2568	1
		1000	2650(2)	8669	192779	30.61	1	(1)	5	(3)	9952	225870	27.45	—	—	—	3259(2)	501	26.66	0	—	—
100	0.025	100	2	778	843	0	3	11	479	3	1070	1372	0	3	20	608	8	87	0	3	22	53
		250	23	1567	6141	0	2	46	26	47	2490	9305	0	2	106	100	134	867	0	2	264	13
		500	162	2348	27594	0	3	203	39	326	3823	42848	0	3	413	43	1611	4214	0	3	1557(1)	5
		100	4	872	1944	0	3	18	233	5	1058	2353	0	3	40	503	12	154	0	3	34	48
		250	152	1634	44208	0	2	557	117	199	2367	49445	0	2	600	217	407	3297	0	2	1043	14
		500	1581	2671	277612	0	2	1806(1)	35	2561	4533	315038	0	2	1591	23	(3)	6651	12.32	—	—	—
	0.05	100	17	865	8679	0	3	123	393	36	1240	11537	0	3	271	507	37	834	0	3	126	68
		250	732	1903	212938	0	2	1440(1)	209	1165	2503	235058	0	2	(2)	148	1618	13780	0	2	(2)	17
		500	(3)	3216	440073	17.68	—	—	—	(3)	4457	305875	22.8	—	—	—	(3)	7803	18.77	—	—	—

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 (3.4), is not feasible with respect to the true distribution, the oracle phase fixes the infeasibility in most cases. 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 any feasibility cuts in 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.

In Table 4.3, 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 [21] to estimate the optimality gaps with statistical guarantees. In particular, we use [21, Theorem 4], 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 4.2, 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 in the "Master" column. In the "Oracle" column we report the minimum and maximum objective function values 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 4.3, 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). The "EGap" column shows 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., the Min value under Oracle), and $\overline{lb}$ is the lower bound obtained from the Min value of Master. Luedtke and Ahmed [21] 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 is valid 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 in the "EGap" column is the estimated optimality gap with approximately 87.5% confidence. We denote with an asterisk the EGap 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 4.3, we observe that the Min/Max values in the "Master" and "Oracle" columns are nondecreasing as the number of scenarios increases. For most of the instances with $(\bar{b}, |\Omega|) = (100, 100)$, both the 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 even provide a feasible solution, as can be seen from Table 4.2, with Inf = 3. Note that even if the 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 4.2 has Inf = 3, which indicates that none of the solutions

provided by the Benders phase are 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 nonunit 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 sample 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, a 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 conclusion, for instances with small $|V_1|$, it is computationally tractable to obtain a truly optimal solution using the exact method Oracle ($\kappa = 2$); see [41]. For larger $|V_1|$, an approximate sampling-based method is more effective than an exact method. Furthermore, solving the sample approximation problem with DCG is more effective than solving the corresponding DEP model. The new valid inequalities enhance the performance of DCG 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 [21] provides provably feasible solutions to the true problem with low optimality gaps at high confidence. Note that we also tested the influence of a larger size of $|V_2|$ compared to $|V_1|$ in [41]. Compared to DCG, DEP 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.

TABLE 4.3
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

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 let $|V_1^2| = n - 10$. 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, (2.5). To solve the PPSC problem under the linear threshold model, we apply three methods. The first method is to solve it exactly by using Algorithm 1, which is referred to as "Oracle." We only study the cases of Oracle with $\kappa = 2$ in this subsection. The second method is the DEP (2.5), which uses the true distribution. We use the default setting of CPLEX with a single thread to solve DEP (2.5). The dynamic programming formulation in DEP (2.5) 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 3.7 and 3.8, and use formulation (3.24) (referred to as DEP-S (3.24)) to solve the sample approximation problem. We summarize the performance of these three methods in Table 4.4. In our experience, the proposed smaller formulation (3.24) has the best performance for our test instances compared to the original DCG-NV (or equivalently DCG-Sub) and the larger DEP (3.4) in (x, y, z)-space for the linear threshold models. So we only report the results with (3.24) for the sample approximation problem. In the case of DEP-S (3.24), 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 for 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 4.4. The "Time" column shows the total time for solving DEP-S (3.24) including the oracle phase. We do not report the solution time for both DEP-S (3.24) and the oracle phase separately since both phases can solve all instances extremely fast. The "Cuts" column of DEP-S (3.24) denotes the number of feasibility cuts (3.3) added to DEP-S (3.24) by the oracle phase. A positive value in "Cuts" indicates that the optimal solution given by DEP-S (3.24) is infeasible with respect to the original problem as detected by the oracle. In Table 4.4, Oracle ($\kappa = 2$) provides solutions for 2 out of 12 the instances. Compared to Oracle, our proposed compact reformulation (DEP (2.5)) solves 9 of the 12 instances optimally under the time limit. Note that both Oracle and DEP (2.5) solve all instances under the true distribution for this dataset. On the other hand, the sampling-based approach, DEP-S (3.24) combined with the oracle phase, provides an approximate solution efficiently. The "nOpt" column shows the number of instances out of three that do not have the same optimal objective value obtained from the exact method. The "oGap" column gives the optimality gap between the optimal value to the sample approximation problem given by DEP-S and the true optimal value given by DEP (2.5) calculated as $100|(v - v^*)|/v$, where v is the objective function value of the feasible solution obtained from DEP-S and v^* is the truly optimal objective function value. We put a "—" in the "oGap" column if the exact method is unable 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 $(|V|, \bar{b}, \epsilon) = (60, 100, 0.0125)$ that has a 2.2% optimality gap. In this case, the sampling-based approach cannot guarantee the optimality even though we use a large number of scenarios. Furthermore, for these linear threshold instances, most solutions to the sample approximation problem are feasible; there are only two settings $((|V|, \bar{b}, \epsilon) = (60, 1, 0.0125)$ and $(60, 100, 0.0125))$ where oracle cuts were necessary to correct the infeasibility of the solution given by the sample approximation.

TABLE 4.4
The exact and sampling methods for PPSC with the linear threshold model.

			Oracle ($\kappa = 2$)		DEP (2.5)	DEP-S (3.24)				
$\|V\|$	$\bar{b}$	ϵ	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	—	—

Our additional experiments with the sampling-based method for $|V| = 120$, summarized in [41], demonstrate the solution quality of the linear threshold instances with $|V| = 120$. To summarize, for the linear threshold instances with $|V| = 120$ and $|\Omega| \geq 500$, DEP-S (3.24) provides a feasible solution efficiently with an estimated optimality gap no larger than 5% with high probability.

5. Conclusions and future work. In this paper, we propose models and methods to solve a probabilistic partial set-covering problem considered in the social networks literature under two 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 the PPSC problem, 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. We note that our decomposition methods are generally applicable to other problems with the desired structure. For example, the probabilistic set-covering problem with a circular distribution considered in [7, 21] fits into our framework, although, in this case, we can also provide a compact MIP using the formulable structure of the probability oracle. For future research, it will be interesting to exploit the structure of the set-covering problems to derive more effective feasibility cuts for the oracle-based exact algorithm.

Acknowledgments. We thank the three referees and the Associate Editor for their constructive comments, which improved the paper. We also thank Baski Balasundaram for bringing [27] to our attention.

REFERENCES

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

[2] W. P. Adams and H. D. Sherali, *Mixed-integer bilinear programming problems*, Math. Program., 59 (1993), pp. 279–305.

[3] S. Ahmed and D. J. Papageorgiou, *Probabilistic set covering with correlations*, Oper. Res., 61 (2013), pp. 438–452.

[4] 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.

[5] R. E. Barlow and K. D. Heidtmann, *Computing k-out-of-n system reliability*, IEEE Trans. Rel., 33 (1984), pp. 322–323.

[6] P. Beraldi and M. E. Bruni, *An exact approach for solving integer problems under probabilistic constraints with random technology matrix*, Ann. Oper. Res., 177 (2010), pp. 127–137.

[7] P. Beraldi and A. Ruszczyński, *The probabilistic set-covering problem*, Oper. Res., 50 (2002), pp. 956–967.

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

[9] G. Ergün, *Human sexual contact network as a bipartite graph*, Phys. A, 308 (2002), pp. 483–488.

[10] M. Fischetti and M. Monaci, *Cutting plane versus compact formulations for uncertain (integer) linear programs*, Math. Program. Comput., 4 (2012), pp. 239–273.

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

[12] R. M. Karp, *Reducibility among combinatorial problems*, in Complexity of Computer Computations, The IBM Research Symposia Series, Springer, Boston, MA, 1972, pp. 85–103.

[13] 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, Association for Computing Machinery, New York, 2003, pp. 137–146.

[14] S. Küçükyavuz, *On mixing sets arising in chance-constrained programming*, Math. Program., 132 (2012), pp. 31–56.

[15] G. Laporte and F. Louveaux, *The integer L-shaped method for stochastic integer programs with complete recourse*, Oper. Res. Lett., 13 (1993), pp. 133–142.

[16] M. Lejeune, *Pattern-based modeling and solution of probabilistically constrained optimization problems*, Oper. Res., 60 (2012), pp. 1356–1372.

[17] X. Liu, F. Kilinç-Karzan, and S. Küçükyavuz, *On intersection of two mixing sets with applications to joint chance-constrained programs*, Math. Program. (2018), https://doi.org/10.1007/s10107-018-1231-2.

[18] X. Liu and S. Küçükyavuz, *A polyhedral study of the static probabilistic lot-sizing problem*, Ann. Oper. Res., 261 (2018), pp. 233–254.

[19] X. Liu, S. Küçükyavuz, and J. Luedtke, *Decomposition algorithms for two-stage chance-constrained programs*, Math. Program., 157 (2016), pp. 219–243.

[20] J. Luedtke, *A branch-and-cut decomposition algorithm for solving chance-constrained mathematical programs with finite support*, Math. Program., 146 (2014), pp. 219–244.

[21] J. Luedtke and S. Ahmed, *A sample approximation approach for optimization with probabilistic constraints*, SIAM J. Optim., 19 (2008), pp. 674–699.

[22] J. Luedtke, S. Ahmed, and G. L. Nemhauser, *An integer programming approach for linear programs with probabilistic constraints*, Math. Program., 122 (2010), pp. 247–272.

[23] G. McCormick, *Computability of global solutions to factorable nonconvex programs. Part* I*: Convex underestimating problems*, Math. Program., 10 (1976), pp. 147–175.

[24] G. Nemhauser and L. Wolsey, *Maximizing submodular set functions: Formulations and analysis of algorithms*, in Studies on Graphs and Discrete Programming, Annals of Discrete Mathematics, Vol. 11, North-Holland Math. Stud. 59, P. Hansen, ed., North-Holland, Amsterdam, 1981, pp. 279–301.

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

[26] A. Nemirovski and A. Shapiro, *Scenario approximations of chance constraints*, in Probabilistic and Randomized Methods for Design under Uncertainty, Springer, London, 2006, pp. 3–47.

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

[28] D. Pollard, *A User's Guide to Measure Theoretic Probability*, Cambridge University Press, Cambridge, 2001.
[29] A. Prékopa, *Contributions to the theory of stochastic programming*, Math. Program., 4 (1973), pp. 202–221.
[30] A. Prékopa, *Dual method for the solution of a one-stage stochastic programming problem with random RHS obeying a discrete probability distribution*, ZOR: Meth. Models Oper. Res., 34 (1990), pp. 441–461.
[31] A. Ruszczyński, *Probabilistic programming with discrete distributions and precedence constrained knapsack polyhedra*, Math. Program., 93 (2002), pp. 195–215.
[32] S. M. Samuels, *On the number of successes in independent trials*, Ann. Math. Statist., 36 (1965), pp. 1272–1278.
[33] A. Saxena, V. Goyal, and M. A. Lejeune, *MIP reformulations of the probabilistic set covering problem*, Math. Program., 121 (2010), pp. 1–31.
[34] Y. Song, *Structure-Exploiting Algorithms for Integer and Chance Constrained Stochastic Programs*, Ph.D. thesis, University of Wisconsin, Madison, 2013.
[35] Y. Song, J. R. Luedtke, and S. Küçükyavuz, *Chance-constrained binary packing problems*, INFORMS J. Comput., 26 (2014), pp. 735–747.
[36] W. van Ackooij, A. Frangioni, and W. Oliveira, *Inexact stabilized Benders' decomposition approaches with application to chance-constrained problems with finite support*, Comput. Optim. Appl., 65 (2016), pp. 637–669.
[37] W. van Ackooij and C. Sagastizábal, *Constrained bundle methods for upper inexact oracles with application to joint chance constrained energy problems*, SIAM J. Optim., 24 (2014), pp. 733–765.
[38] R. van der Hofstad, *Random Graphs and Complex Networks: Volume* 1, Cambridge University Press, Cambridge, 2016.
[39] R. V. Vohra and N. G. Hall, *A probabilistic analysis of the maximal covering location problem*, Discrete Appl. Math., 43 (1993), pp. 175–183.
[40] Y. H. Wang, *On the number of successes in independent trials*, Statist. Sinica, 3 (1993), pp. 295–312.
[41] H.-H. Wu, *Stochastic Combinatorial Optimization with Applications in Graph Covering*, Ph.D. thesis, University of Washington, Seattle, WA, 2018.
[42] H.-H. Wu and S. Küçükyavuz, *A two-stage stochastic programming approach for influence maximization in social networks*, Comput. Optim. Appl., 69 (2018), pp. 563–595.
[43] M. Zhang, S. Küçükyavuz, and S. Goel, *A branch-and-cut method for dynamic decision making under joint chance constraints*, Manag. Sci., 60 (2014), pp. 1317–1333.
[44] 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, Association for Computing Machinery, New York, 2014, pp. 1306–1315.
[45] M. Zhao, K. Huang, and B. Zeng, *A polyhedral study on chance constrained program with random right-hand side*, Math. Program., 166 (2017), pp. 19–64.