Probing-Enhanced Stochastic Programming

Zhichao Ma · Youngdae Kim · Jeff Linderoth · James R. Luedtke · Logan R. Matthews

Received: June, 2024 / Accepted: date

Abstract We consider a two-stage stochastic decision problem where the decision-maker has the opportunity to obtain information about the distribution of the random variables $\boldsymbol{\xi}$ that appear in the problem through a set of discrete actions that we refer to as *probing*. Probing components of a random vector $\boldsymbol{\eta}$ that is jointly-distributed with $\boldsymbol{\xi}$ allows the decision-maker to learn about the conditional distribution of $\boldsymbol{\xi}$ given the observed components of $\boldsymbol{\eta}$. We propose a three-stage optimization model for this problem, where in the first stage some components of $\boldsymbol{\eta}$ are chosen to be observed, and decisions in subsequent stages must be consistent with the obtained information. In the case that $\boldsymbol{\eta}$ and $\boldsymbol{\xi}$ have finite support, Goel and Grossmann gave a mixed-integer programming (MIP) formulation of this problem whose size is proportional to the square of cardinality of the sample space of the random variables. We propose to solve the model using bounds obtained from an information-based relaxation, combined with a branching scheme that enforces the consistency of decisions with observed information. The branch-and-bound approach can naturally be combined with sampling in order to estimate both lower and upper bounds on the optimal solution value and does not require $\boldsymbol{\eta}$ or $\boldsymbol{\xi}$ to have finite support. We conduct a computational study of our method on instances of a stochastic facility location and sizing problem with the option to probe customers to learn about their demands before building facilities. We find that on instances with finite support, our approach scales significantly better than the MIP formulation and also demonstrate that our method can compute statistical bounds on instances with continuous distributions that improve upon the perfect information bounds.

Keywords Stochastic Programming · Decision-Dependent Uncertainty · Branch-and-Bound · Sampling

Zhichao Ma · Jeff Linderoth · Jim Luedtke
Department of Industrial and Systems Engineering, University of Wisconsin-Madison E-mail: zma59@wisc.edu, linderoth@wisc.edu, jim.luedtke@wisc.edu

Jeff Linderoth · Jim Luedtke
Wisconsin Institute for Discovery, University of Wisconsin-Madison

Youngdae Kim · Logan R. Matthews
Energy Sciences, ExxonMobil Technology and Engineering Company

Mathematics Subject Classification (2020) 90C15 · 90C10

1 Introduction

We study optimization models designed to understand the value of obtaining information about the outcome of random variables $\boldsymbol{\xi}$. The models are stochastic optimization problems with a specific but important form of decision-dependent uncertainty. In stochastic programming, the literature on decision-dependent uncertainty typically divides models into two primary categories (Tarhan et al., 2009; Hellemo et al., 2018). In the first category, the probability distribution of $\boldsymbol{\xi}$ is a function of the decision variables, and models in this class are said to have *decision-dependent probabilities*. In the second category, the timing of the revelation of the outcome of $\boldsymbol{\xi}$ is decision-dependent, and models in this class are said to have *decision-dependent information structure*.

Models with decision-dependent information structure are most applicable in a multi-stage setting, where a sequence of decisions are made in stages with the opportunity to observe random outcomes between stages. As pointed out by Dupačová (2006), it is precisely in these contexts where their solution is most challenging. First-stage decisions may influence the marginal and conditional probability distributions in subsequent stages, or if the decisions influence the time when uncertainty is resolved, then the nonanticipativity of future decisions must depend on decisions made at the current stage.

Most of the research on computational approaches for stochastic programs under decision-dependent uncertainty work with a nonanticipative formulation in a setting where the uncertainty is modeled by a finite set of scenarios. The key to this approach is that the collection of nonanticipativity conditions to be enforced depends on the decisions. Goel and Grossmann (2004, 2006) present a MIP (disjunctive) model for this problem, using binary variables to model the time at which each endogenous uncertain parameter is observed. However, the number of (conditional) non-anticipativity constraints form a very large class, roughly on the order of the number of scenarios squared.

Significant subsequent work has been based on improving computational performance of the nonanticapative model by techniques such as Lagrangian relaxation (Goel and Grossmann, 2006; Tarhan et al., 2009), branch-and-cut (Colvin and Maravelias, 2009, 2010), and redundant constraint identification (Boland et al., 2016). A different approach to addressing decision-dependent uncertainty, using linear and piecewise-linear decision rules was given by Vayanos et al. (2011). Mercier and Van Hentenryck (2008) propose a method that is applicable for multi-stage problems with decision-dependent information structure that first applies a sample average approximation (SAA) and then converts that problem to a standard Markov decision process and solves it. Solak et al. (2010) use an SAA method to solve an R&D portfolio management problem, and solve the SAA problem with a Lagrangian relaxation approach.

In our work, we consider a two-stage stochastic programming model as our starting point, where some decisions y are made before the outcome of a random variable $\boldsymbol{\xi}$ is revealed and recourse decisions $z(y, \xi)$ are made after observing ξ. We extend the two-stage model to allow the decision maker to perform a set of discrete actions, which we refer to as *probing*, to reveal information about the distribution

of $\boldsymbol{\xi}$, before making their decisions y. We call this model a *Probing-Enhanced Stochastic Program*. Different from most existing work on stochastic programming with decision-dependent information structure, we do not necessarily assume that taking a probing action gives us complete information about some of the elements of $\boldsymbol{\xi}$. Rather, each candidate probing action gives us the opportunity to observe the value of a component of a different random vector $\boldsymbol{\eta}$, which we assume is correlated with $\boldsymbol{\xi}$, thus giving *information* about the distribution of $\boldsymbol{\xi}$ that may help the decision-maker make a better decision. Our work is related to the discussion of using σ-fields to model the evolution of information described in Artstein (1999) and Artstein and Wets (1993).

Contributions: A primary contribution of our work is to model the problem using a *nested* stochastic programming formulation of the uncertainty, rather than a nonanticiapative one. This allows us to forgo the modeling of conditional non-anticipativity constraints and allows observation (or sampling) of random variables conditional on the outcomes observed by probing. The flexibility afforded by a nested formulation allows for our model to handle random variables $\boldsymbol{\eta}$ arising from a continuous probability distribution.

A second contribution of our work is a branch-and-bound algorithm to solve the nested model that relies on information-based relaxations to generate bounds at nodes of the branch-and-bound tree. The method is exact when the support of the random variables is small enough to allow exact calculation of conditional expectations, and we show how statistical estimates of bounds on the optimal solution value of the nested formulation can be obtained via both internal and external sampling procedures. The external sampling procedure is related to the sample average approximation approach used by Solak et al. (2010), but we use a different method to solve the SAA problem. The internal sampling method can be interpreted as an extension of the stochastic branch-and-bound method of Norkin et al. (1998) to our probing-enhanced model.

We also propose a greedy heuristic for generating quality solutions that efficiently re-uses computations in order to approximate solution quality when comparing potential candidate solutions within the heuristic.

We give computational results that demonstrate that our model can be orders of magnitude faster for the exact solution of small instances with finite support of the random outcomes, when compared to the nonanticaptive formulation, and can improve significantly over the perfect information bound for large instances and instances with continuous distribution. To our knowledge, this is the first computational procedure that is able to significantly improve over perfect-information-based bounds for large-scale problem instances in this class.

Contents: The remainder of the paper is organized as follows. In Section 2, we describe our modeling framework for probing-enhanced stochastic programming, giving both nested and non-anticipative formulations. The nested formulation is posed as a combinatorial optimization problem that selects the set of probing decisions that maximize the expected value, and in Section 3, we describe a branch-and-bound method for its solution that uses information-relaxation based bounds, as well as two different branching methods. We describe in Section 4 two sampling methodologies that may be combined with our branch-and-bound method in both an external and internal manner to obtain statistical upper bounds for the case

when the expected value calculations of the stochastic program cannot be carried out exactly. We also describe how sampling can be used with a candidate probing decision to estimate a statistical lower bound. We describe our greedy heuristic in Section 5 and present results of computational experiments in Section 6.

Notation: We use bold-face math symbols, such as $\boldsymbol{\xi}$, to denote random variables, and light-typeface symbols, such as $\boldsymbol{\xi}(\omega) = \xi$ to denote realizations of random variables. For a random variable $\boldsymbol{\xi}$, the notation $\mathbb{E}_{\boldsymbol{\xi}}[\cdot]$ denotes expectation of the argument with respect to the probability measure associated with the random variable $\boldsymbol{\xi}$. The set of integers $\{1, 2, \ldots, n\}$ is denoted as $[n]$.

2 Background and Models

Our starting point is a classic two-stage stochastic programming problem

$$\max_{y \in Y} \beta^{\mathsf{T}} y + \mathbb{E}_{\boldsymbol{\xi}}[Q(y, \boldsymbol{\xi})], \tag{1}$$

where $\boldsymbol{\xi} : \Omega \to \mathbb{R}^s$ is a s-dimensional random vector defined on a probability space $(\Omega, \mathcal{F}, \mathbb{P})$ having support Ξ, and

$$Y := \{y \in \mathbb{R}^{\hat{p}} \times \mathbb{Z}_+^{p-\hat{p}} \ : \ Ay = b\}$$

is a mixed-integer set in $\mathbb{R}^p$. Given a decision vector $y \in Y$ and outcome $\boldsymbol{\xi}(\omega) = \xi \in \Xi$, the function

$$Q(y, \xi) = \max\{\gamma(\xi)^{\mathsf{T}} z : z \in Z(y, \xi)\} \tag{2}$$

optimizes the recourse decision z, where

$$Z(y, \xi) := \left\{ z \in \mathbb{R}_+^{\hat{q}} \times \mathbb{Z}_+^{q-\hat{q}} : \ T(\xi)y + W(\xi)z = h(\xi) \right\}$$

is a mixed-integer set in $\mathbb{R}^q$ for all $y \in Y$ and $\xi \in \Xi$. We assume that $Q(y, \xi)$ is finite for all $y \in Y$ and $\xi \in \Xi$, which is the standard assumption of *relatively complete recourse* in stochastic programming.

We wish to model a situation in which information about the distribution of $\boldsymbol{\xi}$ can be obtained by taking some discrete actions, which we refer to as *probing*. Abstractly, we model this by considering an n-dimensional random vector $\boldsymbol{\eta} : \Omega \to \mathbb{R}^n$ with support H such that $(\boldsymbol{\eta}, \boldsymbol{\xi})$ is a jointly-distributed random vector defined on the same probability space $(\Omega, \mathcal{F}, \mathbb{P})$.

As a special case of our modeling framework, we may set $\boldsymbol{\eta} \equiv \boldsymbol{\xi}$, in which case the probing decisions correspond to observing individual elements of the random vector $\boldsymbol{\xi}$ in (1). We will use this special case in our numerical experiments, but the methodology described in the paper applies to the general model, with the main computational requirement that samples of $\boldsymbol{\xi}$ conditional on having observed a subset of the entries of $\boldsymbol{\eta}$ can be generated.

We also introduce a *cost-to-probe function* $\alpha : 2^{[n]} \to \mathbb{R}_+$, where $\alpha(S)$ for $S \subseteq [n]$ is the cost of probing the S components of the random vector $\boldsymbol{\eta}$, which we denote as $\boldsymbol{\eta}_S$. For the most part, we assume only that $\alpha(\cdot)$ is a monotone-increasing function; that is, if $S \subseteq T$, then $\alpha(S) \leq \alpha(T)$. One of our branching methods requires the further assumption that $\alpha(\cdot)$ is modular, i.e., $\alpha(S) = \sum_{i \in S} \alpha(\{i\})$ for all subsets S.

2.1 Probing-Enhanced Stochastic Program

In our model, the decision-maker observes the realizations $\boldsymbol{\eta}_S$ before choosing y, so these decisions y can depend on the observed values η_S. This can be advantageous when $\boldsymbol{\xi}$ and $\boldsymbol{\eta}$ are correlated, as the first-stage decisions y may be optimized with respect to the conditional distribution of $\boldsymbol{\xi}$ given the observations η_S.

Define the *conditional-recourse function* analog of (2) given the observations $\boldsymbol{\eta}_S = \eta_S$ for $S \subseteq [n]$ as follows:

$$R(\eta_S) = \max_{y \in Y} \beta^{\mathsf{T}} y + \mathbb{E}_{\boldsymbol{\xi}} \Big[Q(y, \boldsymbol{\xi}) | \boldsymbol{\eta}_S = \eta_S \Big], \tag{3}$$

and for each subset $S \subseteq [n]$ of probing decision, define the *expected conditional-recourse function*

$$F(S) := \mathbb{E}_{\boldsymbol{\eta}_S}[R(\boldsymbol{\eta}_S)].$$

Thus, when we introduce the option of probing, the two-stage stochastic programming problem (1) is extended to the *probing-enhanced stochastic program*

$$z_{\text{PESP}} = \max_{S \subseteq [n]} F(S) - \alpha(S). \tag{PESP}$$

Problem (PESP) seeks to find the optimal set of probing decisions that trades-off the probing cost incurred against the potential gains obtained from knowing more about the distribution of the random variables $\boldsymbol{\xi}$ when making first-stage decisions y.

Calculating the conditional recourse function $R(\eta_S)$ and the expected conditional recourse function $F(S)$ both involve the evaluation of an expectation. In the case that that the random variables have finite support, i.e., both H and Ξ are both finite sets, we can more explicitly detail the computations required to compute $F(S)$.

First, for each $\eta_S \in \text{Proj}_S(\text{H})$, the value of the following finite-support two-stage stochastic program must be calculated:

$$R(\eta_S) = \max_{y \in Y} \beta^{\mathsf{T}} y + \sum_{\xi \in \Xi} \mathbb{P}(\boldsymbol{\xi} = \xi | \boldsymbol{\eta}_S = \eta_S) Q(y, \xi), \tag{4}$$

where $Q(y, \xi)$ is defined in (2). If we define

$$\Xi(\eta_S) := \{\boldsymbol{\xi}(\omega) : \omega \in \Omega, \boldsymbol{\eta}_S(\omega) = \eta_S\} \subseteq \Xi \tag{5}$$

as the outcomes of $\boldsymbol{\xi}$ that are possible after observing η_S, i.e., the scenarios in the conditional distribution, then the evaluation of $R(\eta_S)$ in (4) requires the solution of a two-stage stochastic program with $|\Xi(\eta_S)|$ scenarios. Note that there are $|\,\text{Proj}_S(H)|$ such two-stage stochastic programs to be solved. Given these values, the value of $F(S)$ is then simply calculated as

$$F(S) := \mathbb{E}_{\boldsymbol{\eta}_S}[R(\boldsymbol{\eta}_S)] = \sum_{\eta_S \in \text{Proj}_S(\text{H})} \mathbb{P}(\boldsymbol{\eta}_S = \eta_S) R(\eta_S).$$

In our computations, all two-stage stochastic programs are solved by directly formulating the extensive form stochastic program (Birge and Louveaux, 1997). The evaluation of $F([n])$ requires solving $|\Xi|$ one-scenario problems. The evaluation of $F(\emptyset)$ requires solving one $|\Xi|$-scenario two-stage stochastic program. In general, the smaller the set S, the more computationally challenging the evaluation of $F(S)$ becomes.

2.2 A Non-Anticipativity Based Formulation

In this section, we describe an explicit deterministic equivalent mixed integer programming (MIP) formulation for (PESP), in the case that the support of $\boldsymbol{\eta}$ and $\boldsymbol{\xi}$ is finite, that is due to Goel and Grossmann (2006). A key concept for this formulation is whether two possible scenarios $\eta, \eta' \in \mathrm{H}$ can be distinguished from each other for a given a selection of probing decisions $S \subseteq [n]$.

Definition 1 Given a set $S \subseteq [n]$, scenarios $\eta \in \mathrm{H}$ and $\eta' \in \mathrm{H}$ are *indistinguishable with respect to* S if

$$\sum_{j \in S} |\eta_j - \eta'_j| = 0.$$

We denote the set of all (unordered) pairs of outcomes of $\boldsymbol{\eta}$ as

$$\mathrm{H}^2 := \{\{\eta, \eta'\} \in \mathrm{H} \times \mathrm{H} : \eta \neq \eta'\},$$

and the set of all pairs of random vectors indistinguishable with respect to S as $\mathcal{I}(S) \subseteq \mathrm{H}^2$.

The MIP model starts by writing a non-anticipative formulation of (PESP). If the $S \subseteq [n]$ components of $\boldsymbol{\eta}$ are probed, the decision y can depend (only) on the observed outcome η_S. Let $y(\eta) : \mathrm{H} \mapsto \mathbb{R}^p$ be a mapping corresponding to the first stage decisions y in (1). In order for our decisions $y(\eta)$ to be appropriately nonanticipative (and the mapping $y(\cdot)$ to be measurable), we require that

$$y(\eta) = y(\eta') \quad \mathbb{P} - a.e.\ \{\eta, \eta'\} \in \mathcal{I}(S). \tag{6}$$

That is, if the S components of $\boldsymbol{\eta}$ are probed, and two outcomes $\eta, \eta' \in \mathrm{H}$ are indistinguishable with respect to S, then the same first stage decisions must be made. Note that $\mathcal{I}(\emptyset) = \mathrm{H}^2$, so in this case (6) reduces to ensuring that $y(\eta) = y$ is a constant-valued mapping, and the problem reduces to a standard two-stage stochastic program.

We introduce binary variables $x_j \in \{0, 1\}, j \in [n]$ to indicate whether or not to probe the jth component of $\boldsymbol{\eta}$, and define $S(x) := \{j \in [n] : x_j = 1\}$ as the set of selected probing decisions. We define $z(\xi) : \Xi \mapsto \mathbb{R}^q$ as a measurable mapping corresponding to the second-stage decisions z in (1).

The problem (PESP) can then be formulated as

$$\begin{aligned}
\max_{x,y,z} \quad & \alpha^\mathsf{T} x + \mathbb{E}_{\boldsymbol{\eta},\boldsymbol{\xi}}\left[\beta^\mathsf{T} y(\boldsymbol{\eta}) + \gamma(\boldsymbol{\xi})^\mathsf{T} z(\boldsymbol{\xi})\right] && && \text{(7a)}\\
\text{s.t.} \quad & Ay(\eta) = b && \mathbb{P}\text{-}a.e.\ \eta \in \mathrm{H}, && \text{(7b)}\\
& T(\xi)y(\eta) + W(\xi)z(\xi) = h(\xi) && \mathbb{P}\text{-}a.e.\ (\eta, \xi) \in \mathrm{H} \times \Xi, && \text{(7c)}\\
& y(\eta) = y(\eta') && \mathbb{P}\text{-}a.e.\ (\eta, \eta') \in \mathcal{I}(S(x)), && \text{(7d)}\\
& x \in \{0, 1\}^n && && \text{(7e)}\\
& y(\eta) \in \mathbb{R}^{\hat{p}}_+ \times \mathbb{Z}^{p-\hat{p}}_+ && \mathbb{P}\text{-}a.e.\ \eta \in \mathrm{H}, && \text{(7f)}\\
& z(\xi) \in \mathbb{R}^{\hat{q}}_+ \times \mathbb{Z}^{q-\hat{q}}_+ && \mathbb{P}\text{-}a.e.\ \xi \in \Xi. && \text{(7g)}
\end{aligned}$$

In (7), the nonanticapativity constraints (7d) to be enforced depend on the probing decision vector x. We next discuss how this can be formulated using linear

constraints under the assumption that Y is bounded. Specifically, for every pair of outcomes $(\eta, \eta') \in \mathrm{H}^2$, if the logical implications

$$\sum_{j\in[n]} |\eta_j - \eta'_j| x_j \leq 0 \Rightarrow y_i(\eta) - y_i(\eta') \leq 0 \ \forall i \in [p]$$
$$\sum_{j\in[n]} |\eta_j - \eta'_j| x_j \leq 0 \Rightarrow -y_i(\eta) + y_i(\eta') \leq 0 \ \forall i \in [p]$$

are modeled, then when η and η' are the same on the observed variables, the first stage decision y is forced to take the same values. For each pair of outcomes $\{\eta, \eta'\} \in \mathrm{H}^2$, define

$$\varepsilon(\eta, \eta') := \min_{j\in[n]:|\eta_j - \eta'_j|>0} \{|\eta_j - \eta'_j|\} > 0$$

as the smallest amount by which the two realizations differ. For $i \in [p]$, define

$$R_i := \max_{y\in Y}\{y_i\} - \min_{y\in Y}\{y_i\}$$

as the range of the decision variable y_i. We finally define the big-M value $M_i(\eta, \eta') := [\varepsilon(\eta, \eta')]^{-1} R_i$. With these definitions, we can write a MIP formulation of (PESP) as

$$\max_{x,y,z} \quad \alpha^\mathsf{T} x + \mathbb{E}_{\boldsymbol{\eta},\boldsymbol{\xi}}\left[\beta^\mathsf{T} y(\boldsymbol{\eta}) + \gamma(\boldsymbol{\xi})^\mathsf{T} z(\boldsymbol{\xi})\right] \tag{8a}$$
$$\text{s.t.} \quad Ay(\eta) = b \qquad \mathbb{P}\text{-}a.e.\ \eta \in \mathrm{H}, \tag{8b}$$
$$T(\xi)y(\eta) + W(\xi)z(\xi) = h(\xi) \qquad \mathbb{P}\text{-}a.e.\ (\eta, \xi) \in \mathrm{H} \times \Xi, \tag{8c}$$
$$y_i(\eta) - y_i(\eta') - M_i(\eta, \eta') \sum_{j\in[n]} |\eta_j - \eta'_j| x_j \leq 0 \qquad \mathbb{P}\text{-}a.e.\ \{\eta, \eta'\} \in \mathrm{H}^2, \tag{8d}$$
$$-y_i(\eta) + y_i(\eta') - M_i(\eta, \eta') \sum_{j\in[n]} |\eta_j - \eta'_j| x_j \leq 0 \quad \mathbb{P}\text{-}a.e.\ \{\eta, \eta'\} \in \mathrm{H}^2, \tag{8e}$$
$$x \in \{0,1\}^n \tag{8f}$$
$$y(\eta) \in \mathbb{R}^{\hat{p}}_+ \times \mathbb{Z}^{p-\hat{p}}_+ \qquad \mathbb{P}\text{-}a.e.\ \eta \in \mathrm{H}, \tag{8g}$$
$$z(\xi) \in \mathbb{R}^{\hat{q}}_+ \times \mathbb{Z}^{q-\hat{q}}_+ \qquad \mathbb{P}\text{-}a.e.\ \xi \in \Xi. \tag{8h}$$

If the support of the random variables $\boldsymbol{\eta}$ and $\boldsymbol{\xi}$ is finite, then (8) is a MIP problem with a finite number of constraints and variables. Note, however, that from (8d) and (8e), there are two constraints for every *pair* of distinct outcomes of $\boldsymbol{\eta}$, so this formulation is only tractable if $|\mathrm{H}|$ is quite small. As mentioned in the introduction, some authors such as Boland et al. (2016) have done work to identify a minimal set of nonanticipative constraints (8d) and (8e) that still results in a valid formulation of (PESP), but the number of such constraints may still be extremely large.

3 Branch-and-Bound Algorithms

In this section, we propose two branch-and-bound algorithms to solve (PESP). We describe the algorithms under the assumption that the quantity $F(S)$ can be computed exactly for any $S \subseteq [n]$, for example, when the joint support of $(\boldsymbol{\eta}, \boldsymbol{\xi})$ is finite and not too large. We discuss in Section 4 how sampling can be used to yield statistical approximations when $F(S)$ cannot be computed exactly.

Upper bounds in our branch-and-bound method are based on an information-relaxation concept generalizing the notion of perfect information. Specifically, the relaxations follow from the following observation.

Lemma 1 *If $S \subseteq T \subseteq [n]$, then $F(S) \leq F(T)$.*

Proof Using the definition of $F(T)$ we have

$$\begin{aligned} F(T) &= \mathbb{E}_{\boldsymbol{\eta}_T}\left[\max_{y\in Y} \mathbb{E}_{\boldsymbol{\xi}}\left[\beta^\mathsf{T} y + Q(y,\boldsymbol{\xi})|\boldsymbol{\eta}_T\right]\right] \\ &= \mathbb{E}_{\boldsymbol{\eta}_S}\mathbb{E}_{\boldsymbol{\eta}_{T\setminus S}}\left[\max_{y\in Y} \mathbb{E}_{\boldsymbol{\xi}}\left[\beta^\mathsf{T} y + Q(y,\boldsymbol{\xi})|\boldsymbol{\eta}_S, \boldsymbol{\eta}_{T\setminus S}\right]\right] \\ &\geq \mathbb{E}_{\boldsymbol{\eta}_S}\max_{y\in Y}\mathbb{E}_{\boldsymbol{\eta}_{T\setminus S}}\left[\mathbb{E}_{\boldsymbol{\xi}}\left[\beta^\mathsf{T} y + Q(y,\boldsymbol{\xi})|\boldsymbol{\eta}_S, \boldsymbol{\eta}_{T\setminus S}\right]\right] \\ &= \mathbb{E}_{\boldsymbol{\eta}_S}\left[\max_{y\in Y} \mathbb{E}_{\boldsymbol{\xi}}\left[\beta^\mathsf{T} y + Q(y,\boldsymbol{\xi})|\boldsymbol{\eta}_S\right]\right] = F(S). \end{aligned}$$

□

The set $[n]$ includes the index of all entries in η, so $F([n]) \geq \max\{F(S) : S \subseteq [n]\}$. Recalling that $\alpha : 2^{[n]} \to \mathbb{R}_+$, is a non-negative set function, if we simply drop the term $-\alpha(S)$ in (PESP) we obtain an upper bound on the optimal solution value that is known as the *perfect information bound* $F([n])$:

$$F([n]) \geq z_{\text{PESP}}.$$

On the other hand, probing all components of $\boldsymbol{\eta}$ is one feasible solution to (PESP), which yields the following lower bound on the optimal value:

$$F([n]) - \alpha([n]) \leq z_{\text{PESP}}.$$

3.1 Single-Element Branching

In our first branch-and-bound algorithm, after evaluating the upper and lower bounds at a node, a single component of $j \in [n]$ of $\boldsymbol{\eta}$ is selected, and two new nodes are created, one where $\boldsymbol{\eta}_j$ is probed, and one where $\boldsymbol{\eta}_j$ is *not* probed. Thus, a node P in the branch-and-bound tree is defined by a subset of $S_1^P \subseteq [n]$ of components that are probed and a disjoint subset S_0^P of components that are *not* probed. The following lemma provides lower and upper bounds on the optimal value of the node problem given these restrictions.

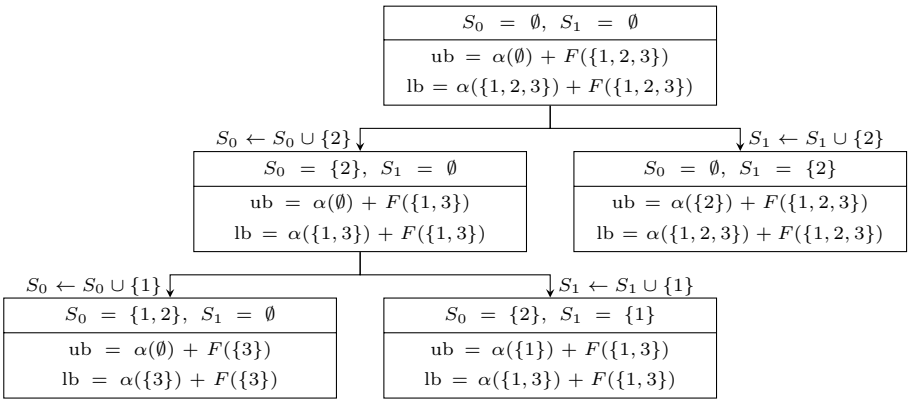


Fig. 1: Partial Bound Calculations for $n = 3$

Lemma 2 *Let $S_0^P, S_1^P \subseteq [n]$ with $S_0^P \cap S_1^P = \emptyset$. Then,*

$$F([n] \setminus S_0^P) - \alpha([n] \setminus S_0^P) \leq \max\{F(S) - \alpha(S) : S \supseteq S_1^P, S \cap S_0^P = \emptyset\} \leq F([n] \setminus S_0^P) - \alpha(S_1^P).$$

Proof The lower bound is obtained by observing that the set $S = [n] \setminus S_0^P$ satisfies $S \supseteq S_1^P$ and $S \cap S_0^P = \emptyset$. The upper bound follows from the monotonicity of α, $\alpha(S_1^P) \leq \alpha(S)$ for any $S \supseteq S_1^P$, and by applying Lemma 1 to conclude that $F([n] \setminus S_0^P) \geq F(S)$ for any S with $S \cap S_0^P = \emptyset$. □

After evaluating node P of the branch-and-bound tree, if its upper bound is not smaller than the value of a known feasible solution, then we must *branch*, by selecting an unfixed component $j \in [n] \setminus S_1^P \setminus S_0^P$, and creating two child nodes P^+ and P^-, with

$$S_1^{P^+} := S_1^P \cup \{j\} \quad S_0^{P^+} := S_0^P,$$
$$S_1^{P^-} := S_1^P \quad S_0^{P^-} := S_0^P \cup \{j\}.$$

The single-element branch-and-bound algorithm is initialized with a root node A defined by $S_0^A = S_1^A = \emptyset$ and then proceeds to create more nodes via the single-element branching procedure. Figure 1 presents an example of a partial branch-and-bound tree with the bound calculations for an instance with $n = 3$. Note that if $S_0^P \cup S_1^P = [n]$ then the lower and upper bounds in Lemma 2 coincide. Thus, there will be at most 2^n leaf nodes in the tree, so the branch-and-bound algorithm is finite. An important aspect of the proposed bounding scheme is that evaluating the bounds for the child node where the probing candidate *must* be probed (it is added to the set $S_1^{P^+}$) requires only an evaluation of $\alpha()$, since all other components of the bounds were computed at the parent node.

Branching variable selection

The choice of branching variable index among the unfixed components $C := [n] \setminus S_1^P \setminus S_0^P$ can significantly impact the size of the search tree, and we would like to select a candidate for which the upper bounds of the two child nodes decrease significantly. For each $j \in C$, we seek to estimate the bound that will be obtained if we create child nodes P^+ and P^- using j as the branching variable. For the node P^+, the computed upper bound changes by

$$\Delta_j^+ := \alpha(S_1^P) - \alpha(S_1^P \cup \{j\}) \leq 0.$$

By evaluating $\alpha(S_1^P \cup \{j\})$ for each unfixed variable $j \in C$, we can decide which unfixed variable will have the most impact for the node P^+. For the node P^-, the computed upper bound changes by

$$\Delta_j^- := F([n] \setminus S_0^P) - F([n] \setminus (S_0^P \cup \{j\})).$$

Thus, we could like to estimate the impact that *not* knowing the outcome of the random variable $\boldsymbol{\eta}_j$ will have on the upper bound, relative to the current upper bound. Letting $S = [n] \setminus S_0^P$, the first term in the definition of Δ_j^- is $F(S) = \mathbb{E}_{\boldsymbol{\eta}_S}[R(\boldsymbol{\eta}_S)]$ and has been calculated (or estimated by sampling) at the current node, but computing $F([n] \setminus (S_0^P \cup \{j\}))$ for each $j \in C$ would be computationally demanding. Thus, we instead seek to use information obtained when evaluating $F(S)$ to estimate the relative importance of knowing each random component $\boldsymbol{\eta}_j$ in terms of its impact on the upper bound. Thus, rather than use Δ_j^- exactly, we use an estimate $\hat{\Delta}_j^-$ and consider two options for this estimate.

The first option is applicable only in the case where each $\boldsymbol{\eta}_j$ is a Bernoulli random variable. In this option, we define $\hat{\Delta}_j^-$ to be an approximation of the value

$$\mathbb{E}_{\boldsymbol{\eta}_S}[R(\boldsymbol{\eta}_S)|\boldsymbol{\eta}_j = 1] - \mathbb{E}_{\boldsymbol{\eta}_S}[R(\boldsymbol{\eta}_S)|\boldsymbol{\eta}_j = 0].$$

In the second option, which applies generally, we define $\hat{\Delta}_j^-$ to be an approximation of the covariance between $\boldsymbol{\eta}_j$ and $R(\boldsymbol{\eta}_S)$. In the case that $F(S)$ is estimated via sampling as described in Section 4, these estimates can be computed using the same information used to evaluate $F(S)$. The details are described in the Appendix.

To combine the Δ_j^+ and $\hat{\Delta}_j^-$ values into a score for each candidate $j \in C$ we first normalize them as follows

$$\zeta_j^+ = \frac{\Delta_j^+ - \min_{j' \in C} \Delta_{j'}^+}{\max_{j' \in C} \Delta_{j'}^+ - \min_{j' \in C} \Delta_{j'}^+}, \quad \zeta_j^- = \frac{\hat{\Delta}_j^- - \min_{j' \in C} \hat{\Delta}_{j'}^-}{\max_{j' \in C} \hat{\Delta}_{j'}^- - \min_{j' \in C} \hat{\Delta}_{j'}^-},$$

then define

$$\psi_j := \zeta_j^+ + \zeta_j^-, \tag{9}$$

and finally choose the branching variable j^* according to $j^* = \mathbf{argmax}_{j \in C} \psi_j$.

3.2 Multi-element branching

In our second branch-and-bound algorithm, we characterize a node P by a subset $S_0^P \subseteq [n]$ of components of $\boldsymbol{\eta}$ that will *not* be probed, and a collection of disjoint subsets $\mathcal{S}^P = \{S_k^P : k \in [n_P]\}$, where for each $k \in [n_P]$ *at least one* of the components of $\boldsymbol{\eta}_{S_k^P}$ will be probed. This method requires the additional assumption that $\alpha(\cdot)$ is modular: $\alpha(S) = \sum_{i \in S} \alpha(\{i\})$ for all $S \subseteq [n]$.

The following lemma provides lower and upper bounds on the optimal value of the node problem given these restrictions.

Lemma 3 *Assume that $\alpha(\cdot)$ is modular. Let $S_0^P \subseteq [n]$, and $\mathcal{S}^P = \{S_k^P : k \in [n_P]\}$ be such that $S_i^P \cap S_j^P = \emptyset \ \forall i \neq j \in [n_P] \times [n_P]$ and $S_0^P \cap S_i^P = \emptyset \ \forall i \in [n_P]$. Then,*

$$\begin{aligned} F([n] \setminus S_0^P) - \alpha([n] \setminus S_0^P) & \\ &\leq \max\{F(S) - \alpha(S) \ : \ S \cap S_0^P = \emptyset, S \cap S_k^P \neq \emptyset, k \in [n_P]\} \\ &\leq F([n] \setminus S_0^P) - \sum_{k \in [n_P]} \min\{\alpha_j : j \in S_k^P\}. \end{aligned}$$

Proof The lower bound is obtained by observing that the set $S = [n] \setminus S_0^P$ satisfies $S \cap S_k^P \neq \emptyset$ for $k \in [n_P]$ and $S \cap S_0^P = \emptyset$. By the monotonicity and modularity of α,

$$\sum_{k \in [n_P]} \min\{\alpha_j : j \in S_k^P\} \leq \alpha(S)$$

for any S with $S \cap S_k^P \neq \emptyset$ for all $k \in [n_P]$. Using Lemma 1 to conclude that $F([n] \setminus S_0^P) \geq F(S)$ for any S with $S \cap S_0^P = \emptyset$ yields the desired upper bound. □

To begin the multi-element branching tree, the root node A is created and evaluated with $S_0^A = \emptyset, \mathcal{S}^A = \emptyset$, which yields the perfect information bound. The first branch creates two nodes L and R enforcing the dichotomy that either no elements of $\boldsymbol{\eta}$ are probed ($S_0^L = [n], \mathcal{S}^L = \emptyset$), or at least one component of $\boldsymbol{\eta}$ is probed, ($S_0^R = \emptyset, \mathcal{S}^R = \{[n]\}$). At all subsequent nodes of the branch-and-bound tree, branching on an active node P is done by selecting a subset element $T \in \mathcal{S}^P$ and a subset $K \subset T$. Since at least one element of T must be selected for probing, we know the following trichotomy:

1. If no elements of K are probed, then at least one element of $T \setminus K$ must be probed;
2. If no elements of $T \setminus K$ are probed, then at least one element of K must be probed;
3. Otherwise at least one element from both T and $T \setminus K$ must be probed.

This translates into three new child nodes P_1, P_2, P_3 of P defined by

1. $S_0^{P_1} = S_0^P \cup K$. $\mathcal{S}^{P_1} = \mathcal{S}^P \setminus \{T\} \cup \{T \setminus K\}$.
2. $S_0^{P_2} = S_0^P \cup (T \setminus K)$. $\mathcal{S}^{P_2} = \mathcal{S}^P \setminus \{T\} \cup \{K\}$.
3. $S_0^{P_3} = S_0^P$. $\mathcal{S}^{P_3} = \mathcal{S}^P \setminus \{T\} \cup \{K\} \cup \{T \setminus K\}$.

Figure 2 presents an example of a partial branch-and-bound tree obtained using multi-element branching for an example with $n = 5$. Compared to single-element branching, this approach tends to lead to subproblems with more elements in the set S_0^P higher in the tree, which may improve the bounds more quickly as the quantity $F([n] \setminus S_0^P)$ is a key component of the node bounds.

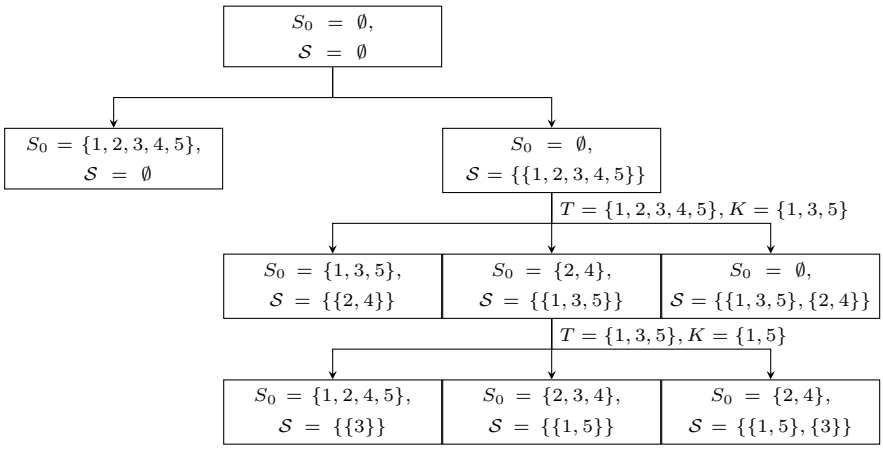


Fig. 2: Example partial branch-and-bound tree with $n = 5$ multi-element branching.

Branching decisions

Branching in the multi-element branching method requires two decisions. A branching set $T \in \mathcal{S}^P$ must be selected, and the elements of T must be partitioned. In our implementation, we select the branching set T from $\mathcal{S}^P$ as the set with the largest cardinality. To partition the elements of the branching set T, we use the importance estimates ψ_j introduced in equation (9). The set is partitioned by choosing K to heuristically minimize the difference $|\sum_{j\in K}\psi_j - \sum_{j\in T\setminus K}\psi_j|$, which tries to ensure that both child nodes are of "equal" importance. Specifically, we sort the elements of T in decreasing order of their scores ψ_j. Let $j_1, j_2, \ldots, j_{|T|}$ be the elements of T sorted such that $\psi_{j_1} \geq \psi_{j_2} \geq \cdots \geq \psi_{j_{|T|}}$. Then we partition T into K and $T \setminus K$ by setting $K = \{j_k : k \text{ is odd}\}$.

4 Sampling

In many applications, the joint support of the random variables $(\boldsymbol{\eta}, \boldsymbol{\xi})$ will be too large allow for the exact evaluation of $F(S)$. We present two methods for estimating an upper bound on z_{PESP} using sampling, which vary according to whether the branch-and-bound algorithm is run on an approximation built using a single sample (external sampling – Section 4.1) or whether sampling is done repeatedly throughout the branch-and-bound algorithm (internal sampling – Section 4.2). We also describe in Section 4.3 how sampling can be used to estimate a lower bound on z_{PESP}.

4.1 External Sampling—Sample Average Approximation

A classical way to obtain bounds on the optimal solution value of a stochastic program is via sample-average approximation (SAA). In SAA, (η^k, ξ^k), for $k \in [N]$ are

jointly randomly sampled according to the distribution $\mathbb{P}$, and an approximation to the problem assuming each of these scenarios is equally likely is created.

Consider a fixed sample and let $S \subseteq [n]$. Let $\hat{\mathrm{H}}_S = \bigcup_{k=1}^{N} \eta_S^k$ be the set of unique subvectors of η^k in the sample. For each $\eta_S \in \hat{\mathrm{H}}_S$, define $\Omega_N(\eta_S) = \{k \in [N] : \eta_S^k = \eta_S\}$. Then, on this random sample, the sample estimate $\mathbf{F}_N(S)$ of $F(S)$ is computed by solving the following two-stage stochastic program for each $\eta_S \in \hat{\mathrm{H}}_S$:

$$\mathbf{R}_N(\eta_S) = \max_{y \in Y} \beta^{\mathsf{T}} y + |\Omega_N(\eta_S)|^{-1} \sum_{k \in \Omega_N(\eta_S)} Q(y, \xi^k) \tag{10}$$

where $Q(y, \xi^k)$ is defined in (2). The value $\mathbf{F}_N(S)$ is then computed as

$$\mathbf{F}_N(S) = N^{-1} \sum_{\eta_S \in \hat{\mathrm{H}}_S} |\Omega_N(\eta_S)| \mathbf{R}_N(\eta_S).$$

Finally, the SAA optimal value for a given sample is defined as

$$\mathbf{v}_N = \max\{\mathbf{F}_N(S) - \alpha(S) : S \subseteq [n]\}. \tag{11}$$

Since $\mathbf{F}_N(S)$ can be computed for any $S \subseteq [n]$ by solving a collection of two-stage stochastic programs with finite support, the sampled approximation problem (11) can be solved using the branch-and-bound method of Section 3. Any method can be used to solve each of the two-stage stochastic programs (10). In our implementation, we use a commercial MIP solver to solve the extensive form Birge and Louveaux (1997), but decomposition methods such as the L-shaped algorithm Van Slyke and Wets (1969); Laporte and Louveaux (1993) or dual decomposition Carøe and Schultz (1999) may also be applied when applicable.

The optimal solution value of the sampled instance $\mathbf{v}_N$ is an outcome of a random variable, as it depends on the randomly drawn sample of size N. It is well-known that the expected value of the random variable $\mathbf{v}_N$ provides a biased estimate of the true objective value: $\mathbb{E}[\mathbf{v}_N] \geq z_{\mathrm{PESP}}$. In order to estimate $\mathbb{E}[\mathbf{v}_N]$, and hence estimate an upper bound of z_{PESP}, we can replicate the computations, as suggested by Mak et al. (1999). Specifically, we draw L independent batches of joint samples $(\eta^{k,\ell}, \xi^{k,\ell})$, for $k \in [N], \ell \in [L]$ from the distribution $\mathbb{P}$. Note that since the sampled elements *within* a batch are not required to be independent, variance reduction techniques such as Latin hypercube sampling (LHS) (McKay et al., 1979; Freimer et al., 2012) could be employed to generate the sample for each fixed batch $\ell \in [L]$. We define $\mathbf{v}_N^\ell$ to be the optimal solution value of (11) coming from the ℓth batch of samples and define the average of the $\mathbf{v}_N^\ell$ values as

$$\boldsymbol{\Upsilon}_{N,L} := L^{-1} \sum_{\ell \in [L]} \mathbf{v}_N^\ell \tag{12}$$

The estimator $\boldsymbol{\Upsilon}_{N,L}$ is an unbiased estimate of $\mathbb{E}[\mathbf{v}_N]$ and by the Central Limit Theorem, we know that

$$\sqrt{L}(\boldsymbol{\Upsilon}_{N,L} - \mathbb{E}[\mathbf{v}_N]) \Rightarrow \mathsf{N}(0, \mathrm{Var}(\mathbf{v}_N)) \text{ as } L \to \infty.$$

The sample estimator of the variance $\mathrm{Var}(\mathbf{v}_N)$ is

$$\mathbf{s}_{N,L}^2 := (L-1)^{-1} \sum_{\ell \in [L]} (\mathbf{v}_N^\ell - \Upsilon_{N,L})^2, \tag{13}$$

which can be used to compute an approximate $(1-\alpha)$ confidence upper bound of $\mathbb{E}[\mathbf{v}_N]$ as

$$\boldsymbol{\Upsilon}_{N,L} + L^{-1/2} t_{L-1,\alpha} \mathbf{s}_{N,L}, \tag{14}$$

where $t_{L-1,\alpha}$ is the α critical value of the Student's t distribution with $L-1$ degrees of freedom. It is also shown in Mak et al. (1999) that $\mathbb{E}\mathbf{v}_N \geq \mathbb{E}\mathbf{v}_{N+1} \geq z_{\text{PESP}}$, so increasing the sample size N reduces the bias of the estimate of z_{PESP}.

Re-using information.

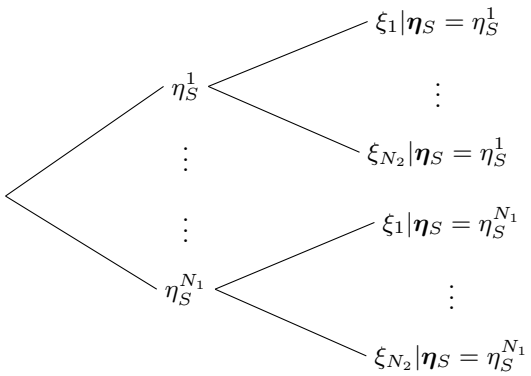
When running the branch-and-bound algorithm with a fixed sample $(\eta^k, \xi^k)_{k=1}^N$ it frequently occurs that the subset $\Omega_N(\eta_S)$ used in the calculation of $\mathbf{R}_N(\eta_S)$ is identical to the subset $\Omega_N(\eta'_{S'})$ for some previously observed $\eta'_{S'} \neq \eta_S$. For example, at nodes in which the upper bound is computed based on a set S that consists of many elements of $[n]$ (as often happens in early nodes in the branch-and-bound tree) the sets $\Omega_N(\eta_S)$ have small cardinality, or may even be singletons, and hence repeat often. Thus, we suggest storing the computed values of $\mathbf{R}_N(\eta_S)$ in a hash table with the key defined by a binary encoding of the set $\Omega_N(\eta_S)$. Thus, any time we need to evaluate $\mathbf{R}_N(\eta_S)$ we first determine the set $\Omega_N(\eta_S)$ and check this hash table to see if the the required two-stage stochastic program has already been solved. In Section 6.5, we report on the significant computational savings that occur from this simple observation.

4.2 Internal Sampling

A limitation of the external sampling approach is that within a single replication, the same sample is used to estimate $F(S)$ for all S that are encountered within the branch-and-bound algorithm, which prevents taking advantage of the specific subset of observed elements of $\boldsymbol{\eta}_S$ when estimating $F(S)$. In particular, if the support of $\boldsymbol{\eta}_S$ is infinite then the set $\hat{\mathrm{H}}_S$ of unique vectors η_S^k in any randomly-drawn finite sample will include the full sample so that the set $\Omega_N(\eta_S)$ is a singleton for each $\eta_S \in \hat{\mathrm{H}}_S$. This implies that the estimate $F_N(S)$ will revert to the perfect information bound for *any finite sample* N, making it impossible for the external sampling approach to improve upon the perfect information bound. Even if the support of $\boldsymbol{\eta}_S$ is finite but very large, the same reasoning suggests that the bias of the lower bound $\mathbf{v}_N$ used in the external sampling approach may be very large.

To overcome this drawback, sampling can be done separately to estimate an upper bound on each node within the branch-and-bound search. This approach is referred to as *internal* sampling because the sampling is done internal to the branch-and-bound search, and can be considered an adaptation of the stochastic branch-and-bound algorithm of Norkin et al. (1998) to our probing enhanced stochastic programming model. The advantage of internal sampling in this setting is that we can then do *conditional* sampling in a nested fashion. To estimate bounds at a node P within the branch-and-bound algorithm we require an estimate of an upper bound of $F([n] \setminus S_0^P)$. A lower bound on the optimal value is obtained by estimating a lower bound of $F(S)$ for a candidate solution S. We discuss these cases separately, and also discuss how to obtain an estimate of a global upper bound from a branch-and-bound tree.

Fig. 3: Internal Sampling



4.2.1 Statistical Upper Bounds on $F(S)$

Consider a node in a branch-and-bound tree with associated non-probed set S_0^P and define $S = [n] \setminus S_0^P$. The main computational task is to estimate a statistical upper bound on $F(S)$.

Let $\eta_S^k, k \in [N_1]$ be a sample of $\boldsymbol{\eta}_S$, and for each observation η_S^k, let $\xi^{ki}, i \in [N_2]$ be a *conditional* sample of $\boldsymbol{\xi}$ taken from the conditional distribution of $\boldsymbol{\xi}$ given $\boldsymbol{\eta}_S = \eta_S^k$. Figure 3 illustrates this nested sampling procedure.

An estimate of $F(S)$ is then obtained as

$$\bar{\mathbf{F}}_{N_1,N_2}(S) = N_1^{-1} \sum_{k \in [N_1]} \bar{\mathbf{R}}_{N_2}(\eta_S^k) \tag{15}$$

where

$$\bar{\mathbf{R}}_{N_2}(\eta_S^k) = \max_{y \in Y} \beta^\top y + N_2^{-1} \sum_{i \in [N_2]} Q(y, \xi^{ki}).$$

Note that computing this estimate $\bar{\mathbf{F}}_{N_1,N_2}(S)$ requires solving N_1 two-stage stochastic programs, each with a sample of size N_2

The following result demonstrates that this point estimate of $F(S)$ can provide the basis of a statistical upper bound estimate.

Lemma 4 *Assume that $\eta^1, \ldots, \eta^{N_1}$ are identically distributed and for each $k \in [N_2]$ the sample $\xi^{k1}, \ldots, \xi^{kN_2}$ are identically distributed conditional on $\boldsymbol{\eta}_S = \eta_S^k$. Then,*

$$\mathbb{E}_{\eta^{N_1}, \xi^{N_2}}[\bar{\mathbf{F}}_{N_1,N_2}(S)] \geq F(S),$$

where $\mathbb{E}_{\eta^{N_1}, \xi^{N_2}}$ indicates the expectation is taken with respect to the samples.

Proof Based on the bias result in sample average approximation (Mak et al., 1999), for each $k \in [N_1]$ it holds that

$$\mathbb{E}_{\xi^{N_2}}\left[\bar{\mathbf{R}}_{N_2}(\eta_S^k)\right] \geq R(\eta_S^k)$$

where the expectation is taken with respect to the sample $\xi^{k1}, \ldots, \xi^{kN_2}$ that is conditional on $\boldsymbol{\eta}_S = \eta_S^k$. Thus,

$$\begin{aligned}\mathbb{E}_{\eta^{N_1}, \xi^{N_2}}\left[\bar{\mathbf{F}}_{N_1,N_2}(S)\right] &= \mathbb{E}_{\eta^{N_1}}\left[N_1^{-1}\sum_{k\in[N_1]}\mathbb{E}_{\xi^{N_2}}[\bar{\mathbf{R}}_{N_2}(\eta_S^k)]\right] \\ &\geq \mathbb{E}_{\eta^{N_1}}\left[N_1^{-1}\sum_{k\in[N_1]}R(\eta_S^k)\right] \\ &= \mathbb{E}_{\boldsymbol{\eta}_S}[R(\boldsymbol{\eta}_S)] = F(S).\end{aligned}$$

□

If the sample $\eta^1, \ldots \eta^{N_1}$ is *independent* and identically distributed, then the values $\bar{\mathbf{R}}_{N_2}(\eta_s^k)$ for $k \in [N_1]$ are iid and hence can be used to compute an approximate statistical upper bound for $F(S)$. By the Central Limit Theorem, we have

$$\sqrt{N_1}\Big(\bar{\mathbf{F}}_{N_1,N_2}(S) - \mathbb{E}\big[\bar{\mathbf{R}}_{N_2}(\boldsymbol{\eta}_S)\big]\Big) \to N\big(0, \mathrm{Var}(\bar{\mathbf{R}}_{N_2}(\boldsymbol{\eta}_S)\big),$$

as $N_1 \to \infty$. The sample variance of $\bar{\mathbf{R}}_{N_2}(\boldsymbol{\eta}_S)$ is computed as

$$\mathbf{s}^2_{R_{N_2}}(N_1) = (N_1 - 1)^{-1}\sum_{k\in[N_1]}\big(\bar{\mathbf{R}}_{N_2}(\eta_S^k) - \bar{\mathbf{F}}_{N_1,N_2}\big)^2.$$

Thus, for N_1 sufficiently large, we can approximate the probability that $F(S)$ exceeds a constant u with the probability that a normal random variable with mean $\bar{\mathbf{F}}_{N_1,N_2}(S)$ and standard deviation $\mathbf{s}^2_{R_{N_2}}(N_1)$ exceeds u.

Variance reduction techniques such as LHS can be used to improve these estimates similarly as was described for external sampling in Section 4.1. This can be accomplished by creating the sample $\eta^1, \ldots \eta^{N_1}$ as a set of *batches* where the samples in each batch are generated via LHS, but the batches are generated independently. Then, if the number of batches is sufficiently large, the sample variance of the batch averages is used in place of $\mathbf{s}^2_{R_{N_2}}(N_1)$ for the upper bound estimate.

For purposes of incorporating these upper bounds with the branch-and-bound algorithm of Section 3, it is crucial to note that these bounds are *statistical* in nature. Specifically, if the bounds are used to fathom nodes, there is a chance the optimal solution to the problem would be excluded in the search. Thus, in our branch-and-bound implementation of internal sampling, we *never* fathom a node by bound and simply enumerate nodes until a specified time limit is reached. In our implementation, we always select to evaluate the node with the largest (statistical) upper bound, emulating the well-known *best-bound* node selection rule from MIP (Linderoth and Savelsbergh, 1999).

Exploiting small support of $\boldsymbol{\eta}$ or $\boldsymbol{\xi}$

If the random variable $\boldsymbol{\xi}$ has finite support (e.g., if each element is a Bernoulli random variable) there may be situations where the conditional support of $\boldsymbol{\xi}$ given a sample of η_S^k of $\boldsymbol{\eta}_S$ is small enough that $R(\eta_S^k)$ can be computed exactly as described in (4). In this case, one can replace the sample estimate $\bar{\mathbf{R}}_{N_2}(\eta_S^k)$ with its true evaluation $R(\eta_S^k)$ in (15) to obtain a lower variance estimator, and the remainder of the upper bound derivation is unchanged.

On the other hand, when the support of $\boldsymbol{\eta}_S$ is small (e.g., in case elements of $\boldsymbol{\eta}$ are independent Bernoulli trials and $|S|$ is small), we can use this structure to obtain a potentially lower variance estimate of $F(S)$ by summing over these observations. In this situation, let $\eta_S^1, \ldots, \eta_S^{N_S}$ be the set of all possible observations of $\boldsymbol{\eta}_S$, and let $p(\eta_S^k)$ be the probability of observing outcome η_S^k. For each $k \in [N_S]$, we take M independent batches, each of size N_2, of conditional samples of ξ given $\boldsymbol{\eta}_S = \eta_S^k$, say ξ^{kij} for $i \in [M], j \in [N_2]$. For each $k \in [N_1]$ and each batch $i \in [M]$, let

$$\hat{\mathbf{R}}_{k,i,N_2}(\eta_S^k) = \max_{y \in Y} \beta^\top y + N_2^{-1} \sum_{j \in [N_2]} Q(y, \xi^{kij})$$

and let $\mu_{k,N_2}(\eta_S^k) = \mathbb{E}[\hat{\mathbf{R}}_{k,i,N_2}]$ be the expected value of each batch. Then, for each $k \in [N_1]$ define the average of the batch values as

$$\bar{\mathbf{R}}_{k,M,N_2}(\eta_S^k) = M^{-1} \sum_{i \in [M]} \hat{\mathbf{R}}_{k,i,N_2}(\eta_S^k).$$

The estimate $\bar{\mathbf{R}}_{k,M,N_2}(\eta_S^k)$ is an unbiased estimator of $\mu_{kN_2}(\eta_S^k)$. When the M batch samples are i.i.d., by the Central Limit Theorem, we have that as $M \to \infty$

$$\sqrt{M}[\bar{\mathbf{R}}_{k,M,N_2}(\eta_S^k) - \mu_{k,N_2}(\eta_S^k)]] \to N\Big(0, \operatorname{Var}\big(\hat{\mathbf{R}}_{k,i,N_2}(\eta_S^k)\big)\Big). \tag{16}$$

The sample variance of $\hat{\mathbf{R}}_{k,i,N_2}(\eta_S^k)$ is computed as

$$\mathbf{s}^2_{R_k}(M) = (M-1)^{-1} \sum_{i \in [M]} \big(\hat{\mathbf{R}}_{k,i,N_2}(\eta_S^k) - \bar{\mathbf{R}}_{k,M,N_2}(\eta_S^k)\big)^2.$$

Define

$$\begin{aligned} \bar{F}(S) &:= \sum_{k \in [N_S]} p(\eta_S^k) \mathbb{E}[\bar{\mathbf{R}}_{N_2}(\eta_S^k)] \text{ and} \\ \mathbf{F}_{M,N_2}(S) &:= \sum_{k \in [N_S]} p(\eta_S^k) \bar{\mathbf{R}}_{k,M,N_2}(\eta_S^k). \end{aligned} \tag{17}$$

Observe that

$$\bar{F}(S) \geq \sum_{k \in [N_S]} p(\eta_S^k) R(\eta_S^k) = F(S).$$

Therefore, $\bar{F}(S)$ is an upper bound of $F(S)$, and $\mathbf{F}_{M,N_2}(S)$ is an unbiased estimator of $\bar{F}(S)$ and we can use it as a statistical upper bound of $F(S)$. Aggregating (16) for $k \in [N_S]$ yields

$$\sqrt{M}[\mathbf{F}_{M,N_2}(S) - \bar{F}(S)] \to N\Big(0, \sum_{k \in [N_S]} p(\eta_S^k)^2 \operatorname{Var}\big(\hat{\mathbf{R}}_{k,i,N_2}(\eta_S^k)\big)\Big)$$

as $M \to \infty$. Thus, for M sufficiently large, we can approximate the probability that $F(S)$ exceeds a constant u with the probability that a normal random variable with mean $\mathbf{F}_{M,N_2}(S)$ and standard deviation $\mathbf{s}_M(S)$ exceeds u, where

$$\mathbf{s}_M(S) = \Big(\sum_{k \in [N_S]} p(\eta_S^k)^2 \mathbf{s}^2_{R_k}(M)\Big)^{1/2}.$$

4.2.2 Estimating a Global Upper Bound

Now consider a branch-and-bound tree with set of leaf nodes $\mathcal{P}$, where for each node $P \in \mathcal{P}$ the information-based upper bound is calculated as $c^P + F([n] \setminus S_0^P)$, where the constant c^P is determined via either Lemma 2 or Lemma 3, depending on the branching mechanism used. Thus, the best upper bound on z_{PESP} that can be obtained from this tree is the value:

$$\max\{c^P + F([n] \setminus S_0^P) : P \in \mathcal{P}\}.$$

Using the upper bound procedure described in Section 4.2.1, we obtain an upper bound estimate $\mathbf{U}_P$ of $F([n] \setminus S_0^P)$ for each $P \in \mathcal{P}$ which is approximately normally distributed with an estimated mean, say μ_P, and estimated standard deviation, say s_P. Then, a value u is a global upper bound if and only if $\mathbf{U}_P \leq u$ for all $P \in \mathcal{P}$, and hence, using independence of the estimates for each node $P \in \mathcal{P}$, we can compute a $1-\alpha$ confident upper bound on z_{PESP} as

$$u(\mathcal{P}) = \inf_{u \in \mathbf{R}} \Big\{ u : \prod_{P \in \mathcal{P}} \mathbb{P}(\mathbf{U}_P \leq u) \geq 1 - \alpha \Big\}.$$

Using the normal approximations, $u(\mathcal{P})$ can be found via

$$u(\mathcal{P}) = \inf_{u \in \mathbf{R}} \Big\{ u : \prod_{P \in \mathcal{P}} \Phi\Big((u - \mu_P)/s_P\Big) \geq 1 - \alpha \Big\}, \tag{18}$$

which can be estimated by binary search.

In our description of the branch-and-bound algorithm using internal sampling we assume that for each node we use a fixed sample to estimate the upper bound, so that the estimated upper bound at a node is only improved by branching on that node. While we do not explore this here since our focus in this paper is on deriving upper bound and branching techniques for the probing structure, we mention that one can also consider variations where the upper bound at leaf nodes is improved by doing more sampling, and the sampling effort is strategically allocated to leaf nodes – see, e.g., Norkin et al. (1998); Xu and Nelson (2013).

4.3 Statistical Lower Bounds on $F(S)$ and z_{PESP}

Any candidate set of probing actions $S \subseteq [n]$ defines a feasible solution of (PESP), and hence

$$z_{\text{PESP}} \geq F(S) - \alpha(S).$$

Thus, a statistical lower bound on z_{PESP} can be obtained by estimating a statistical lower bound on $F(S)$ for any S.

Let $\eta_S^k, k \in [N_1]$ be a sample of $\boldsymbol{\eta}_S$, and for each observation η_S^k, choose a solution $\hat{y}^k \in Y$. For each $k \in [N_2]$, let $\hat{\xi}^{ki}, i \in [N_2]$ be a sample of $\boldsymbol{\xi}$ taken from the conditional distribution of $\boldsymbol{\xi}$ given $\boldsymbol{\eta}_S = \eta_S^k$. Then, an estimate of $F(S)$ is obtained as

$$\underline{\mathbf{F}}_{N_1,N_2}(S) = N_1^{-1} \sum_{k \in [N_1]} \underline{\mathbf{R}}_{N_2}(\eta_S^k) \tag{19}$$

where

$$\underline{\mathbf{R}}_{N_2}(\eta_S^k) = \beta^\top \hat{y}^k + N_2^{-1} \sum_{i\in[N_2]} Q(\hat{y}^k, \hat{\xi}^{ki}).$$

Computing this estimate requires solving $N_1 \cdot N_2$ recourse problems to evaluate $Q(\hat{y}^k, \hat{\xi}^{ki})$ for each $k \in [N_1]$ and $i \in [N_2]$.

As shown in Lemma 5, $\underline{\mathbf{F}}_{N_1,N_2}(S)$ provides a statistical lower bound regardless of the choice of $\hat{y}^k \in Y$ for $k \in [N_1]$. However, the quality of this lower bound will be influenced by how good the solution $\hat{y}^k$ is to the problem defined by $R(\eta_S^k)$. Thus, we suggest to choose $\hat{y}^k$ by taking a separate conditional sample of $\boldsymbol{\xi}$ given $\boldsymbol{\eta}_S = \eta_S^k$, $\xi^{ik}, i \in [N_3]$ for each $k \in [N_1]$ and choosing

$$\hat{y}^k \in \arg\max_{y\in Y} \beta^\top y + N_3^{-1} \sum_{i\in[N_3]} Q(y, \xi^{ki}). \tag{20}$$

Computing $\hat{y}^k$ for $k \in N_1$ in this way requires solving N_1 two-stage stochastic programs, each having N_3 scenarios.

Lemma 5 *Let $\hat{y}^k \in Y$ for $k \in [N_1]$. Assume that $\eta^1, \ldots, \eta^{N_1}$ are identically distributed and for each $k \in [N_2]$ the sample $\xi^{k1}, \ldots, \xi^{kN_2}$ is identically distributed conditional on $\boldsymbol{\eta}_S = \eta_S^k$. Then,*

$$\mathbb{E}_{\eta^{N_1},\xi^{N_2}}[\underline{\mathbf{F}}_{N_1,N_2}(S)] \le F(S).$$

Proof For each $k \in [N_1]$, $\hat{y}^k \in Y$ implies that

$$R(\eta_S^k) \ge \beta^\top \hat{y}^k + \mathbb{E}_{\boldsymbol{\xi}}\big[Q(\hat{y}^k, \hat{\xi}) \,\big|\boldsymbol{\eta}_S = \eta_S^k\big].$$

Thus,

$$\begin{aligned}
\mathbb{E}_{\eta^{N_1},\xi^{N_2}}\big[\underline{\mathbf{F}}_{N_1,N_2}(S)\big] &= \mathbb{E}_{\eta^{N_1}}\Big[N_1^{-1} \sum_{k\in[N_1]} \mathbb{E}_{\xi^{N_2}}[\underline{\mathbf{R}}_{N_2}(\eta_S^k)]\Big] \\
&= \mathbb{E}_{\eta^{N_1}}\Big[N_1^{-1} \sum_{k\in[N_1]} \mathbb{E}_{\xi^{N_2}}\big[\beta^\top \hat{y}^k + N_2^{-1} \sum_{i\in[N_2]} Q(\hat{y}^k, \hat{\xi}^{ki})\big]\Big] \\
&= \mathbb{E}_{\eta^{N_1}}\Big[N_1^{-1} \sum_{k\in[N_1]} \Big(\beta^\top \hat{y}^k + \mathbb{E}_{\boldsymbol{\xi}}\big[Q(\hat{y}^k, \boldsymbol{\xi})|\boldsymbol{\eta}_S = \eta_S^k\big]\Big)\Big] \\
&\le \mathbb{E}_{\eta^{N_1}}\Big[N_1^{-1} \sum_{k\in[N_1]} R(\eta_S^k)\Big] = \mathbb{E}_{\boldsymbol{\eta}_S}[R(\boldsymbol{\eta}_S)] = F(S).
\end{aligned}$$

□

If the sample $\eta^1, \ldots \eta^{N_1}$ is *independent* and identically distributed, then the values $\underline{\mathbf{R}}_{N_2}(\eta_S^k)$ for $k \in [N_2]$ are iid and hence can be used to compute an approximate statistical lower bound for $F(S)$. By the Central Limit Theorem, we have

$$\sqrt{N_1}\Big(\underline{\mathbf{F}}_{N_1,N_2}(S) - \mathbb{E}\big[\underline{\mathbf{R}}_{N_2}(\boldsymbol{\eta}_S)\big]\Big) \to N\Big(0, \mathrm{Var}\big(\underline{\mathbf{R}}_{N_2}(\boldsymbol{\eta}_S)\big)\Big)$$

as $N_1 \to \infty$. The sample variance of $\underline{R}_{N_2}(\boldsymbol{\eta}_S)$ is computed as

$$\mathbf{s}^2_{\underline{R}_{N_2}} = (N_1 - 1)^{-1} \sum_{k\in[N_1]} (\underline{\mathbf{R}}_{N_2}(\eta_S^k) - \underline{\mathbf{F}}_{N_1,N_2}(S))^2.$$

Let $t_{N_1-1,\alpha}$ be the critical value of Student's t distribution. Then, we obtain the $1-\alpha$ confidence lower bound estimate of $F(S)$ as

$$\underline{\mathbf{F}}_{N_1,N_2}(S) - t_{N_1-1,\alpha}\mathbf{s}_{\underline{R}_{N_2}} N_1^{-1/2}.$$

5 Greedy Heuristic

At every node in the branch-and-bound method of Section 3 there is a natural candidate solution that can be used to obtain a valid (statistical) lower bound on the optimal solution value via the method described in Section 4.3. Moreover, in this case, there is no need to solve the auxiliary two-stage stochastic programs described in (20), since the evaluation of $F(S)$ naturally provides high-quality candidate solutions y^k. However, the solution is obtained by paying for probing for *all* components of $\boldsymbol{\eta}$ that are not fixed at the node. Hence, especially at the top of the branch-and-bound tree, the feasible solutions probe many components of $\boldsymbol{\eta}$, and the solutions may be far from optimal. In this section, we describe a greedy heuristic to obtain a feasible solution to (PESP). While the greedy method is very natural, our contribution is to identify techniques for relatively efficiently evaluating candidate probing actions to greedily add to the current solution at each iteration.

We first state in Algorithm 1 the greedy heuristic in its natural form, which is not practical to implement due to the need to evaluate (or even estimate) $F(S)$ for a large number of sets S. We use the notation $S^c = [n] \setminus S$. The method begins with $S = \emptyset$ as the initial solution. In each iteration we have a solution S, and for each element $j \in S^c$, we evaluate the solution obtained by adding that element to S, and choose one that gives the largest value. As it is possible that the solutions obtained may improve or decline at each iteration, the method continues until all elements are probed and stores all selected solutions in the set $\mathcal{L}$, at the end returning the best solution obtained.

Algorithm 1: Naive greedy heuristic.

1 $S \leftarrow \emptyset$. $\mathcal{L} \leftarrow \{S\}$
2 **repeat**
3 | Evaluate $F(S)$.
4 | **for** $j \in S^c$ **do**
5 | | $z_j \leftarrow \alpha(S \cup \{j\}) + F(S \cup \{j\})$
6 | **end**
7 | Choose $j^* \in \arg\max\{z_j : j \in S^c\}$
8 | $S \leftarrow S \cup \{j^*\}$, $\mathcal{L} \leftarrow \mathcal{L} \cup S$.
9 **until** $S = [n]$;
10 **return** $\arg\max\{F(S) - \alpha(S) : S \in \mathcal{L}\}$.

The naive greedy algorithm requires $O(n^2)$ evaluations of $F(S)$, which is impractical unless the joint support of $(\boldsymbol{\eta}, \boldsymbol{\xi})$ is small. We thus employ the conditional sampling method of Section 4.2 to estimate $F(S)$. However, even doing this estimation $O(n^2)$ times is computationally prohibitive for a heuristic.

In our proposed heuristic, each time we obtain a new solution S, the evaluation of $F(S)$ (line 3 of the naive greedy heuristic) is conducted using a slight adaptation of the sampling procedure for estimating a lower bound described in Section 4.3. Just as in Section 4.3, we first take a sample $\eta_S^1, \ldots, \eta_S^{N_1}$ of $\boldsymbol{\eta}_S$, and for each $k \in [N_1]$ we obtain a solution $\hat{y}^k \in Y$ by solving (20) using a conditional sample $\xi^{k1}, \ldots, \xi^{kN_3}$ of $\boldsymbol{\xi}$ conditional on $\boldsymbol{\eta}_S = \eta_S^k$. At this point, the evaluation diverges slightly in that we next take a *joint sample* $(\eta_{S^c}^{k1}, \xi^{k1}), \ldots, (\eta_{S^c}^{kN_2}, \xi^{kN_2})$ of $(\boldsymbol{\eta}_{S^c}, \boldsymbol{\xi})$

conditional on $\boldsymbol{\eta}_S = \eta_S^k$ for each $k \in [N_1]$. This joint sample will be used to facilitate a faster estimation of $F(S \cup \{j\})$ for each $j \in S^c$ when evaluating the next element to add to the set in the greedy method. Next, we let $\hat{Y} = \{\hat{y}^1, \ldots, \hat{y}^{N_1}\}$ be the observed solutions and compute the values $Q(\hat{y}^k, \xi^{ki})$ for all $k \in [N_1]$, and $i \in [N_2]$, which requires solving $N_1 \times N_2$ recourse problems. Then, the heuristic estimates $F(S)$ by

$$N_1^{-1} \sum_{k \in [N_1]} \max_{y \in \hat{Y}} \Big\{ \beta^\top y + N_2^{-1} \sum_{i \in [N_2]} Q(y, \hat{\xi}^{ki}) \Big\}. \tag{21}$$

Note that the set Y is replaced by the limited set of solutions $\hat{Y}$, so that this quantity can be computed without solving any additional optimization problems.

An important aspect of our heuristic is the ability to reuse computations to quickly estimate $F(S \cup \{j\}$ for $j \in S^c$. We next fix $j \in S^c$ and discuss how we estimate $F(S \cup \{j\})$. For each $k \in [N_1]$, we use K-means clustering to partition the (scalar) values $\{\eta_j^{ki} : i \in [N_2]\}$ into K sets of similar values, and let the scenarios in these sets be $\Omega_{N_2}^{k\ell}$ for $\ell \in [K]$, so that these sets are disjoint and $\bigcup_{\ell \in [K]} \Omega_{N_2}^{k\ell} = [N_2]$. Our approximation is based on the assumption that taking probing action j will allow us to distinguish between scenarios in different sets $\Omega_{N_2}^{k\ell}$, but not scenarios within each of these sets. We also continue to use the set $\hat{Y}$ in place of Y, and thus estimate the value of $F(S \cup \{j\})$ as

$$N_1^{-1} \sum_{k \in [N_1]} K^{-1} \sum_{\ell \in [K]} \max_{y \in \hat{Y}} \Big\{ \beta^\top y + |\Omega_{N_2}^{k\ell}|^{-1} \sum_{i \in \Omega_{N_2}^{k\ell}} Q(y, \hat{\xi}^{ki}) \Big\}. \tag{22}$$

The key observation is that $Q(y, \xi^{ki})$ has already been computed for all $y \in \hat{Y}$, $k \in [N_1]$, and $i \in [N_2]$, and hence computing (22) does not require solving any additional optimization problems.

Thus, our proposed greedy heuristic follows the structure of Algorithm 1, with the differences being that the evaluation of $F(S)$ in line 3 is replaced by the approximation (21) and the evaluation of $F(S \cup \{j\})$ in line 5 is replaced by (22).

The computational effort of the greedy heuristic is impacted heavily by the choice of the sample sizes N_1 and N_2. To generate solutions relatively quickly, one may use relatively small sample sizes – e.g., we use $N_1 = 20$, $N_2 = 20$, and $N_3 = 50$ in our experiments. While smaller sample sizes may be sufficient for guiding the greedy search, it is important to evaluate the most promising solutions using larger samples to estimate a lower bound as described in Section 4.3. In our implementation, we use the estimated solution values obtained by (21) within the heuristic to select the ten most promising solutions from the set of solutions $\mathcal{L}$, and then evaluate these selected solutions with larger sample sizes.

6 Computational Results

In this section, we first introduce a prototype application. Then we present the results of a series of computational experiments that demonstrate the impact of various algorithmic choices and the effectiveness of our branch-and-bound algorithm.

6.1 Probing-Enhanced Facility Location

Our sample application is an extension of a two-stage stochastic facility location problem. There is a set of potential facilities I, and each of the facilities $i \in I$ may be built in one of a given set of (capacity) configurations C_i. There are also a given set of customers J, and each customer $j \in J$ has a (random) demand of $\mathbf{d}_j$. Customers may only have their demand served from a single facility. The objective of the problem is to select a set of facilities to open, to configure the facilities, and to assign customers to facilities in order to maximize the expected profit. There is a (fixed) cost β^{c}_{ic} of opening facility $i \in I$ in configuration $c \in C_i$ and a fixed cost β^{a}_{ij} of assigning customer $j \in J$ to facility $i \in I$. These assignments must be done before observing the customer demands. There are binary variables y_{ic} that take the value 1 if and only if facility i is opened in configuration $c \in C_i$ and binary variables u_{ij} that take value 1 if and only if customer $j \in J$ is assigned to facility $i \in I$. The stochastic facility location problem can then be written as

$$\max_{y,u} \; -\sum_{i\in I}\sum_{c\in C_i} \beta^{\mathrm{c}}_{ic} y_{ic} - \sum_{i\in I}\sum_{j\in J} \beta^{\mathrm{a}}_{ij} u_{ij} + \mathbb{E}_{\mathbf{d}}[Q(y,u,\mathbf{d})] \tag{23a}$$

$$\text{s.t.} \sum_{c\in C_i} y_{ic} \le 1 \qquad \forall i \in I, \tag{23b}$$

$$u_{ij} - \sum_{c\in C_i} y_{ic} \le 0 \qquad \forall i \in I, \forall j \in J, \tag{23c}$$

$$\sum_{i\in I} u_{ij} \le 1 \qquad \forall j \in J, \tag{23d}$$

$$y_{ic} \in \{0,1\} \qquad \forall i \in I, \forall c \in C_i, \tag{23e}$$

$$u_{ij} \in \{0,1\} \qquad \forall i \in I, \forall j \in J. \tag{23f}$$

Constraints (23b) and (23c) ensure that at most one configuration is selected for each facility and that customers are assigned only to open facilities. Constraint (23d) ensures that customers are only assigned to one facility. In (23a), the term $\mathbb{E}_{\mathbf{d}} Q(y,u,\mathbf{d})$ is the expected revenue obtained when opening and sizing facilities according to y and assigning customers to facilities according to u. For a given realization of demands $\mathbf{d}(\omega) = d$ and first-stage solution y and u, the maximum revenue can be obtained by solving the following linear program:

$$Q(y,u,d) := \max_{f} \quad \sum_{i\in I}\sum_{j\in J} r f_{ij} \tag{24a}$$

$$\text{s.t.} \sum_{j\in J} f_{ij} - \sum_{c\in C_i} \theta_{ic} y_{ic} \le 0 \qquad \forall i \in I, \tag{24b}$$

$$f_{ij} - d_j u_{ij} \le 0 \qquad \forall i \in I, \forall j \in J, \tag{24c}$$

$$f_{ij} \ge 0 \qquad \forall i \in J, \forall j \in J. \tag{24d}$$

The decision variable f_{ij} is the number of units of product that customer $j \in J$ received from facility i, and each unit of customer demand that is met results in a revenue of r. The parameters θ_{ic} are the production capacity of facility i if operated in configuration $c \in C_i$.

6.1.1 Definition of Uncertainty

In our computational experiments, we have families of instances with two different distributions of customer demands. In the first instances, the customer demands are independent random variables following a discrete distribution with two outcomes—either the customer demand is 0 with probability ρ_j, or it is a known nominal value η_j with probability $(1 - \rho_j)$. Thus, instances in this family have a total of $2^{|J|}$ scenarios.

In the second family, customer demands are continuous random variables. Each customer j can either have a low demand or a high demand. The probability of a low demand for customer j is ρ_j, and the probability of a high demand is $(1 - \rho_j)$. For a low demand, the demand follows a triangular distribution with a minimum value of 0, a maximum value of η_1, and a mode of 0. For a high demand, the demand follows a triangular distribution with a minimum value of η_2, mode η_3, and maximum value of η_4. The demand for each customer is independent. Instances are available from the authors in JSON format upon request.

6.1.2 Information Model

The probing-enhanced stochastic programming model introduction in Section 2 allows for an arbitrary correlation structure between $\boldsymbol{\eta}$ and $\boldsymbol{\xi}$. In this proof-of-concept implementation, we assume that $\boldsymbol{\eta} = \boldsymbol{\xi}$. In the context of our facility location model, this implies that by probing customer $j \in J$, we can exactly know that customer's true demand realization. However, because our stochastic model assumes the customer demands are independent, this gives no information about the distribution of demand of *other* customers. This information structure is equivalent to earlier work on *decision-dependent information structure* that models the ability to control the *timing* of when the outcome of random variables is known. Here, if we probe customer $j \in J$, then we know its demand before deciding facility locations, sizes, and customer assignment to open facilities. Future work will consider different models of correlation between $\boldsymbol{\eta}$ and $\boldsymbol{\xi}$.

6.2 Computing Environment and Test Instances

All two-stage stochastic programs are solved via the extensive form (Birge and Louveaux, 1997) and these and any other optimization problems are solved with Gurobi v9.5.1. A computational advantage of the branch-and-bound methods proposed in Section 3 when combined with sampling is that independent estimations of problem bounds can be computed in parallel. Most of the computations in this section were carried out on a shared cluster of machines of varying architectures scheduled with the HTCondor software (Thain et al., 2005). Thus, in order to report computational effort and compare different approaches we rely primarily on Gurobi's *work unit* (GWU), which in our experience is a more reliable statistic to use for comparison between runs on different machines. According to Gurobi's documentation, a GWU is very roughly equivalent to a second of CPU runtime, but this may vary significantly depending on the hardware. Unless otherwise specified, we enforce a total computational budget of 30000 GWU for obtaining an upper bound using our branch-and-bound methods. We also report

other machine-independent statistics of computational effort, such as the total nodes of all branch-and-bound trees, the total number of MIP problems solved, etc. For computations done in parallel, we report statistics aggregated over all distributed computations.

Most of the computational effort in the information-relaxation-based branch-and-bound method is on the computation or estimation of $F(S)$ for different subsets $S \subseteq [n]$ throughout the branch-and-bound tree. Recall from the description of the single-element branching method of Section 3.1 that the value of $F(S)$ needs to be re-computed for only one of the two child nodes. Similarly, the value of $F(S)$ needs to be recomputed for only two of the three child nodes in the multi-element branching method of Section 3.2. So in some computational tables, we report both the number of nodes in the tree and the number function evaluations of $F(S)$ required.

We do a majority of our testing on twelve instances, six with discrete demand distribution and six with continuous distribution as described in Section 6.1.1 All instances have $|I| = 5$ facilities, and each facility has $|C_i| = 4$ possible configurations. The name of each instance encodes the number of customers, with an instance whose name starts with "Jn" having $n = |J|$ customers, and instances whose names end in "_C" have customers whose demands follow a continuous distribution.

Table 1: Comparison of Nonanticipative Formulation and Information-Relaxation Based Branch-and-Bound Algorithm

	NA-Formulation			**Info-Relax B&B**		
Instance	**# Nodes**	**GWU**	**Gap (%)**	**# Nodes**	**# Eval**	**GWU**
J4	1	1.5	0.0	11	8	0.1
J5	235	173.0	0.0	18	14	0.2
J6	170	5411.6	0.0	47	39	1.8
J7	1	30000.0	39.7	44	37	4.9
J8	1	30000.0	191.2	89	74	31.1
J9	-	-	-	101	81	84.4
J10	-	-	-	276	234	363.7
J20_1	1	30000.0	30.7	85920	75050	1801.7
J20_2	1	30000.0	35.2	40399	35234	652.3
J20_3	1	30000.0	7.4	72893	63304	586.5

6.3 Comparison of Nonanticipative and Nested Formulations

We first report results of experiments designed to compare the performance of a standard branch-and-cut-based MIP solver on the nonanticipativity-based formulation (8) with the information-relaxation-based branch-and-bound algorithm for solving the probing-enhanced facility location problem. We perform two types of comparisons. First, we exactly solve small instances having between 4 and 10 customers and a discrete demand distribution by both methods. Second, we solve sampled instances (with sample size $N = 100$) that have 20 customers by both methods. The multi-way branching method describe in Section 3.2 is used for the

information-relaxation-based scheme. Table 1 shows the results of these experiments, where # Nodes is the number of nodes explored either within the work unit limit or until terminated, GWU is the Gurobi work units, Gap (%) is the ending optimality gap reported by the MIP solver when the work limit was reached, and # Eval is the number of nodes in the information-based branch-and-bound algorithm on which the $F(S)$ needed to be evaluated.

For instances J9 and J10, the nonanticapative formulation could not be created due to a lack of memory. The information-relaxation-based branch-and-bound method was able to solve all instances to optimality within the work-unit limit. The table indicates that the information-relaxation-based branch-and-bound method can outperform the direction solution of the nonanticipativity-based formulation by several orders of magnitude.

6.4 Impact of Variance Reduction in Sampling Methods

The quality of the statistical estimates of the solution bounds described in Section 4 depend heavily on the variance of observations. In Table 2, we demonstrate the significant reduction in variance in the estimates that can be obtained by using Latin hypercube sampling (LHS). The table shows for the instance J20_3 the estimate of $\mathbb{E}[\mathbf{v}_N]$ for two different values of N and the standard deviation of the estimate. There were $L = 30$ replications used to to compute the estimates. The estimate $\Upsilon_{N,L}$ was computed by equation (12), and the standard deviation of the estimate of the mean $s_{N,L}$, was computed by equation (13). The table clearly shows a significant variance reduction when sampling via LHS for these instances, so all remaining computational results presented use LHS for obtaining estimates.

Table 2: Sampling Method Test on Instance J20_3

Sampling	N	$\Upsilon_{N,L}$	$s_{N,L}$
MC	50	12431.8	964.9
LHS	50	12527.8	110.9
MC	100	12464.6	540.2
LHS	100	12368.0	78.7

6.5 Impact of Storing $R_N(\eta_S)$ Evaluations

Recall from the description of using the branch-and-bound method to solve an externally-sampled instance in Section 4.1 that the value of the same two-stage stochastic program computed as $R_N(\eta_S)$ in (10) may be required at many different nodes in the branch-and-bound tree. In Table 3, we show how often the values are reused for our discrete instances with $|J| = 20$ customers, externally sampled with a sample size of $N = 50$, and replicated $L = 30$ times. Our implementation uses a hash table which (for memory purposes) stores only the most recently used $10,000$ values of $R_N(\eta_S)$. For these instances, we show the average GWU, the average number of nodes in the branch-and-bound trees (# Nodes), the average

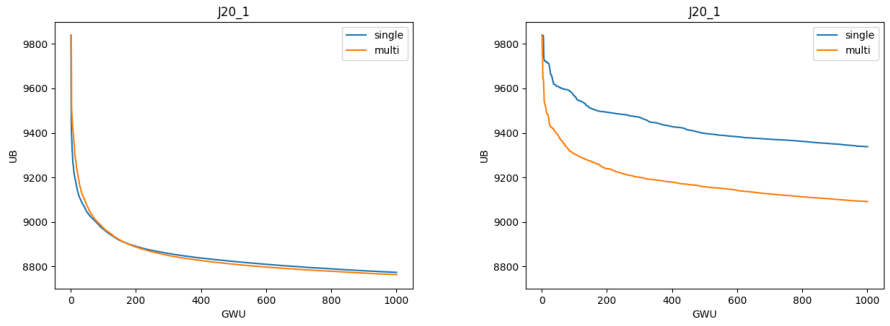
number of function evaluations of $F(S)$ that are required (# Eval.), the average number of two-stage stochastic programs (values $R_N(\eta_S)$) that are required for the computation (#SP), and the percentage of time the value of $R_N(\eta_S)$ can be recovered from a stored value (% Lookup). We show this data for both the single-variable branching method of Section 3.1 and multi-variable branching of Section 3.2.

Table 3: Impact of Hashing

Instance	Method	GWU	# Nodes	# Eval.	# SP	% Lookup
J20_1	single	110.8	116332.6	83017.7	3531655.8	99.79
J20_1	multi	112.4	32770.9	29181.9	1174542.4	99.35
J20_2	single	111.9	78279.1	56509.7	2493461.7	99.81
J20_2	multi	117.3	23871.6	21187.1	895986.7	99.44
J20_3	single	52.2	82441.4	58852.5	2585136.0	99.83
J20_3	multi	65.4	29537.0	26030.0	1098195.1	99.50

Table 3 shows that a remarkably high percentage of the values $R_N(\eta_S)$ can be reused during the course of the branch-and-bound search. The percentage is uniformly larger for the single-variable branching method, which is consistent with the intuition that the child node subproblems change less under single-variable branching, and if a subproblem changes less, then it is more likely to be able to reuse computations when re-evaluating its bound.

Figure 4 displays this phenomenon graphically, depicting the evolution of the upper bound of the branch-and-bound tree as a function of the GWU for an externally-sampled ($N = 100$) instance of J20_1 for both single-variable and multi-variable branching rules. The figure demonstrates that the bound evolution is very similar when hashing is used, but significantly better for the multi-variable branching rule if the values of $R(\eta_S)$ are not stored and re-used. However, using hashing leads to significantly faster reduction of the upper bound in both cases.



(a) Use Hash Table to Look Up Values

(b) Recompute All Values

Fig. 4: Evolution of upper bound for externally-sampled instance of J20_1 instance with $N = 100$

6.6 Comparison of Branching Methods

This subsection is aimed at providing an empirical comparison between the single-element branching method of Section 3.1 and the multi-element branching method of Section 3.2.

External Sampling

As explained at the beginning of Section 4.2, employing the external sampling method for instances with a continuous probability distribution will yield only the perfect-information bound. Thus, to evaluate branching methods for external sampling, we consider only our test instances that come from a discrete probability distribution.

In the experiment, we run the information-relaxation-based branch-and-bound method of Section 3 on sampled instances of sizes $N \in \{50, 100, 200\}$ with different branching rules and compare the performance. We employ $L = 30$ different joint samples of $(\boldsymbol{\eta}, \boldsymbol{\xi})$ for each value of N to estimate $\mathbb{E}[\mathbf{v}_N]$ via equation (12). Each replication is run with a GWU limit of 1000.

We test three branching methods in this experiment. Method `random` follows the single variable branching method presented in Section 3.1 with the choice of probing element to branch on chosen randomly among the candidates. Method `single` also does single variable branching, but uses the first option described in the *branching variable selection* portion of Section 3.1 for selecting the probing element to branch on. Specifically, we use the observed difference between values of $R(\boldsymbol{\eta}_S)$ for the two outcomes for probing customer j, $(\mathbb{E}_{\boldsymbol{\eta}_S}[R(\boldsymbol{\eta}_S)|\boldsymbol{\eta}_j = 1] - \mathbb{E}_{\boldsymbol{\eta}_S}[R(\boldsymbol{\eta}_S)|\boldsymbol{\eta}_j = 0])$, for the measure of importance of excluding $j \in [n]$ from probing consideration, $\hat{\Delta}_j^-$. Method `multi` follows the multi-element branching scheme described in Section 3.2, with branching decisions made using the same scoring as in method `single`.

Table 4 shows the results of this experiment. This table displays, for each combination of instance, sample size (N), and branching method employed: the number of the 30 replications that are successfully solved within the work unit limit (Solved), the average number of nodes solved over the $L = 30$ replications (#Nodes), the average number of evaluations of the function $F(S)$ that are required (#Eval), the average number of work units (GWU), and the estimate of the upper bound $\boldsymbol{\Upsilon}_{N,L}$ (UB). We do not test method `rand` on the instances with $N = 200$ because the instances with $N \in \{50, 100\}$ were sufficient to demonstrate the superior performance of `single` over `rand`.

Since for each of the three branching methods, all $L = 30$ of the sampled instances having $N = 50$ solved successfully, the estimated value of $\mathbb{E}[\mathbf{v}_{50}]$ (in UB) is the same in all cases. In case that one or more of the 30 replications did not solve, we use upper bound of the branch-and-bound tree at the work limit as the value $\mathbf{v}_N^\ell$ in equation (12) to obtain a valid statistical upper bound on z_{PESP}.

From Table 4 we conclude from comparing the `random` and `single` branching methods that our strategy for selecting the branching entity described significantly outperforms branching randomly. We next observe that there is little difference in performance between single and multi-variable branching in terms of the total number of work units required to solve an instance or the bound obtained in a fixed number (1000) of work units. Note, however, that the number of nodes and function

Table 4: Branching Method Performance for Externally Sampled Instances

Instance	N	**Method**	**Solved**	**# Nodes**	**# Eval.**	**GWU**	**95% UB**
J20_1	50	random	30	124479.0	89333.8	256.8	9012.6
J20_1	50	single	30	116332.6	83017.7	110.8	9012.6
J20_1	50	multi	30	32770.9	29181.9	112.4	9012.6
J20_2	50	random	30	86553.5	62983.4	212.0	10785.5
J20_2	50	single	30	78279.1	56509.7	111.9	10785.5
J20_2	50	multi	30	23871.6	21187.1	117.3	10785.5
J20_3	50	random	30	99418.4	71926.8	121.1	12563.9
J20_3	50	single	30	82441.4	58852.5	52.2	12563.9
J20_3	50	multi	30	29537.0	26030.0	65.4	12563.9
J20_1	100	random	0	78585.1	55584.7	1000.1	8999.1
J20_1	100	single	0	162637.2	112977.8	1000.1	8890.3
J20_1	100	multi	0	63584.7	55793.2	1000.0	8865.7
J20_2	100	random	0	74902.0	53348.6	1000.0	10726.5
J20_2	100	single	8	147508.7	102501.2	937.6	10632.6
J20_2	100	multi	10	54807.4	47906.7	943.9	10627.2
J20_3	100	random	0	119891.7	84635.9	1000.0	12425.7
J20_3	100	single	25	161778.2	111920.8	645.5	12358.5
J20_3	100	multi	28	68844.8	59858.6	668.8	12356.2
J20_1	200	single	0	59605.0	41213.0	1000.0	8934.9
J20_1	200	multi	0	16636.9	14489.3	1000.0	8891.9
J20_2	200	single	0	52628.9	36447.0	1000.0	10689.6
J20_2	200	multi	0	14464.6	12522.2	1000.0	10671.5
J20_3	200	single	0	80465.0	55133.2	1000.0	12364.1
J20_3	200	multi	0	25446.3	21803.1	1000.0	12373.0

evaluations required are significantly larger for single-variable compared to multi-variable branching. To explain this apparent contradiction, the reader is reminded of the results of Section 6.5, where it is demonstrated that single variable branching allows for significantly more reuse of portions of the $F(S)$ computations. A second, more subtle reason for faster evaluation of nodes in single-variable branching is that the number of excluded candidate probing actions at nodes by multiple-variable branching is larger than for single-node branching. As mentioned in Section 2.1, the difficulty of evaluating $F(S)$ depends inversely on the cardinality of S, so as more candidates probing actions are excluded from consideration, the upper bound calculation of $F([n] \setminus S_0^P)$ tends to require more computational effort for branch-and-bound nodes in a multiple-variable branching tree.

Internal Sampling

Table 5 shows the results of an experiment comparing the single and multiple variable branch-and-bound methods on instances solved via our internal-sampling approach.

Our internal sampling implementation uses LHS to generate the external sample of size $N_1 = 300$ as 30 independent batches of size 10. The conditional sample has size $N_2 = 100$, and the statistical upper bound is estimated via equation (15). If, however, the cardinality of the support of $\boldsymbol{\eta}_S$ is less than or equal to 8, we forgo the sample and enumerate all possible outcomes, as explained at the end of subsection 4.2.1. In this case, for each possible η_S^k, we take $M = 30$ independent batches of conditional samples ξ given $\boldsymbol{\eta}_S = \eta_S^k$ of size $N_2 = 100$ and we estimate a

Table 5: Branching Method Performance for Internally Sampled Instances

Instance	Method	# Nodes	# Eval.	95% UB
J20_1	single	844	547	8940.1
J20_1	multi	275	218	8834.1
J20_2	single	529	343	10607.7
J20_2	multi	124	94	10611.5
J20_3	single	1260	838	12076.8
J20_3	multi	358	283	12283.4
J25_1	single	847	539	9446.4
J25_1	multi	214	165	9502.2
J25_2	single	602	393	11091.0
J25_2	multi	158	121	11172.5
J25_3	single	342	225	10553.9
J25_3	multi	178	135	10372.6
J20_1_C	single	223	149	10496.5
J20_1_C	multi	167	127	10401.4
J20_2_C	single	183	122	10709.1
J20_2_C	multi	127	98	10584.9
J20_3_C	single	193	127	11423.6
J20_3_C	multi	137	104	11279.8
J25_1_C	single	141	93	11688.8
J25_1_C	multi	86	64	11560.1
J25_2_C	single	153	101	10147.7
J25_2_C	multi	104	78	10064.7
J25_3_C	single	119	81	9231.3
J25_3_C	multi	68	51	9093.6

statistical upper bound on $F(S)$ using equation (17). When selecting a branching candidate, we use the observed sample covariance between $\boldsymbol{\eta}_j$ and $R(\boldsymbol{\eta}_S)$ (from equation (25)) for the importance score $\hat{\Delta}_j$.

All instances in the table were run with a maximum GWU limit of 30000. The table shows the number of nodes of the branch-and-bound tree (#Nodes), the number of evaluations of $F(S)$ (# Eval), and the 95% confident estimate of an upper bound on z_{PESP} (95% UB) computed by (18) obtained by the method within the work limit by both branching methods.

For instances whose random variables are discrete, again the performance difference between single and multiple variable branching seems negligible in the internal sampling method. Note that single-variable branching is able to evaluate significantly many more nodes within the work limit than multiple-variable branching. However, as the sampling is conditional in the internal-sampling approach, there is no ability to reuse portions of the $F(S)$ computations. Thus, the reason for the similarity in performance between the methods is different for internal sampling than it is for external sampling. In the case of internal sampling, the difference is explained by the fact that in single-variable branching, nodes P tend to have fewer candidate probing actions excluded, so the estimation of $F([n] \setminus S_0^P)$ is somewhat easier. Also, there are more nodes whose bounds can be estimated using the ability to exploit the small support of the random variable $\boldsymbol{\eta}_S$, as explained at the end of Section 4.2.1.

For continuous distribution instances, the performance of the multiple-variable branching seems slightly better than the single-variable branching.

6.7 Comparison of External and Internal Sampling

We next compare the performance of external and internal sampling methods in terms of the quality of upper bounds they are able to obtain within the same work limit. The runs (and all parameters values) used to make this comparison are the same as used for the experiments described in Section 6.6, and we use the multi-variable branching method. For the internal sampling method we use a single branch-and-bound search with total work limit of 30000 GWU, and for the external sampling method we use $L = 30$ replications, each with a work limit of 1000 GWU. For external sampling, the 95% confidence interval was computed by (14), and for internal sampling, the confidence 95% confidence estimate for the upper bound for internal sampling is found by equation (18).

In Table 6, for the discrete distribution test instances we present the 95% confidence upper bound on z_{PESP} obtained by the internal sampling method and the external sampling method with three different sample sizes $N \in \{50, 100, 200\}$. We also present the 95% confidence upper bound on z_{PESP} obtained by the internal sampling on the continuous distribution instances, and for context, the table includes the point estimates of the perfect information bound and the best lower bound for each instance. Most of the best solutions were found by the greedy heuristic of Section 5, and we provide more details of lower bound computations in Section 6.8. Note that the values of the PI Bound and the Best LB are point estimates of the values, while we present a 95% confidence limit for the upper bounds obtained from branch-and-bound. The standard error associated with the perfect information and lower bound estimates is in the range of $50 - 100$ and is given in Table 9 in the appendix.

Table 6: 95% Confidence-Level Upper Bound Obtained by External and Internal Sampling Methods

	External			**Internal**	**PI**	**Best**
Instance	$N = 50$	$N = 100$	$N = 200$		**Bound**	**LB**
J20_1	9048.6	8881.5	8903.9	8834.1	9739.2	7919.0
J20_2	10807.6	10645.6	10683.3	10611.5	11465.6	9551.0
J20_3	12597.6	12378.1	12386.8	12283.4	13565.2	11454.0
J25_1	9702.3	9697.1	9744.2	9502.2	10404.7	8278.5
J25_2	11370.6	11330.7	11401.7	11172.5	12576.6	9756.4
J25_3	10668.6	10605.8	10671.2	10372.6	11389.8	8882.9
J20_1_C				10401.5	11287.9	9278.4
J20_2_C				10584.9	11631.4	9302.1
J20_3_C				11279.8	12411.7	10015.2
J25_1_C				11560.1	12330.1	9971.6
J25_2_C				10064.7	10801.0	8787.1
J25_3_C				9093.6	9796.7	7552.6

From Table 6 we observe that the internal sampling method gives moderately better upper bounds than the external sampling method within this fixed work limit on the discrete instances. Second, comparing the change in upper bound obtained by the external sampling method as the sample size N increases can provide an indication of the reduction in bias of the estimate of z_{PESP} as N increases. If all externally-sampled instances were able to be solved to optimality

within the 1000 GWU limit, we would expect the bounds to be monotonically-decreasing. (Refer to Table 4 to see how many of the 30 instances were solved to optimality). Thus, we find that the upper bounds from $N = 100$ are generally lower than those obtained with $N = 50$, but this trend reverses when increasing N to 200, due to the time limit being reached more often on the $N = 200$ instances. Finally, we find that the information-relaxation-based branch-and-bound method significantly improves over the upper bound obtained from the perfect-information relaxation. We are unaware of other algorithms for this problem class that can yield similar improvement.

6.8 Performance of Greedy Heuristic

Finally, we study the computational performance of the greedy heuristic method of Section 5 by comparing the value of the best solution found by the heuristic compared to other methods that more directly rely on solutions obtained by the branch-and-bound search.

In our implementation of the greedy heuristic, at each iteration we solve $N_1 = 20$ two-stage stochastic programs, each having $N_3 = 50$ scenarios, to get our restricted set of solutions Y. We use the same $N_1 = 20$ samples as the outer sample, and for each $\eta_S^k, k \in [N_1]$ in the sample, we take a conditional sample of size $N_2 = 20$ to estimate the value of $F(S)$ at step 3 of the heuristic given in Algorithm 1. To estimate $F(S \cup \{j\})$ at step 5 of the algorithm, we use K-means clustering with $K = 4$, as described in Section 5.

The value of each feasible solution, regardless of how it was generated, was estimated using the process described in Section 4.3. Given a candidate probing set S, we take sample of size $N_1 = 25$ of $\boldsymbol{\eta}_S$, and for each η_S^k in the sample we obtain a solution $\hat{y}^k, k \in [N_1]$, by solving a two-stage stochastic program (20) with $N_3 = 100$ scenarios of $\boldsymbol{\xi}$ generated conditionally on $\boldsymbol{\eta}_S = \eta_S^k$. Then for each $k \in [N_1]$, an independent conditional sample of $\boldsymbol{\xi}$ of size $N_2 = 2000$ is taken and the recourse problems $Q(\hat{y}^k, \hat{\xi}^{ki})$ are solved for $k \in [N_1]$ and $i \in [N_2]$ to obtain the lower bound via equation (19).

Table 7 shows the estimated values of the best solution found by the greedy heuristic method, the best solution obtained from running the internal sampling branch-and-bound method, and the best solution found during *any* of the replications of the externally-sampled branch-and-bound trees. As the values were all estimated using the sample sample sizes N_1, N_2 and N_3, all estimates have approximately the same standard error, which tends to be around 50.

When evaluating the results of Table 7, it is important to take into consideration that the value is the *best* observed from solutions in that category, and the different methods had different numbers of candidate solutions. The greedy heuristic method estimates solution value for the ten solutions whose initial estimates are best. For the internal sampling approach, only the solution coming from the leaf node P with the largest estimated lower bound value of $F([n] \setminus S_0^P) - \alpha([n] \setminus S_0^P)$ was evaluated. For the externally-sampled approach, the best solution from each of the 30 replications for each of the branching methods and for each sample sizes $N \in \{50, 100, 200\}$ is re-evaluated. (Thus, there are 240 potential solutions for the J20 instances and 180 potential solutions for the J25 instances).

Table 7: Estimated Objective Values of Best Solutions Found

Instance	Heuristic	Internal	External
J20_1	7859.8	7734.6	7808.4
J20_2	9511.8	9484.9	9551.0
J20_3	11353.6	11219.6	11280.1
J25_1	8199.0	8212.8	8076.3
J25_2	9654.7	9451.7	9638.4
J25_3	8804.7	8751.3	8836.7
J20_1_C	9278.3	9019.2	-
J20_2_C	9224.7	9136.7	-
J20_3_C	10015.2	9926.3	-
J25_1_C	9971.6	9807.7	-
J25_2_C	8732.4	8486.4	-
J25_3_C	7549.6	7455.0	-

A primary takeaway from the comparison between solution methods is that the greedy heuristic performs quite well at finding high-quality solutions compared to other methods, especially when taking into account the difference in number of different solutions whose value was estimated. This takeaway is reinforced by Figure 5 which shows histograms of the estimated objective values of the solutions for the 10 solutions evaluated by the heuristic and all the solutions found by external sampling, for two instances with discrete distribution and 25 customers.

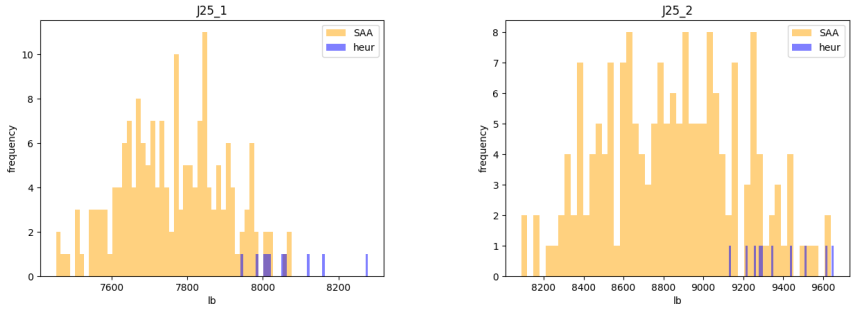


Fig. 5: Distribution of Solution Values for J25_1 and J25_2

The total computational effort for the heuristic is summarized in Table 8. The table shows the total number of GWU required to perform the entire greedy heuristic (Heur), and the total GWU required to accurately estimate the value of all 10 of the perceived best solutions from the greedy procedure (Eval). The table demonstrates that the computational effort of the heuristic is quite modest in comparison to the work limit imposed in the branch-and-bound methods. The effort to perform a more accurate evaluation of the candidate solutions is more significant, but this effort is required, however, for obtaining an unbiased estimate of the objective value from a solution obtained by any method.

Table 8: Work Units Required for the Heuristic

Instance	Heur.	Eval.
J20_1	170.1	1735.7
J20_2	279.1	5547.6
J20_3	139.9	1847.4
J25_1	373.7	4867.5
J25_2	339.7	4998.5
J25_3	521.4	9392.6
J20_1_C	357.0	5589.5
J20_2_C	466.0	6179.1
J20_3_C	426.5	6235.2
J25_1_C	856.8	12123.8
J25_2_C	795.8	11245.9
J25_3_C	1229.4	35910.2

7 Conclusions

We have introduced the *probing-enhanced stochastic program*, a new paradigm for modeling the decision-dependence of the distribution of random variables in stochastic programming. We develop an information-relaxation-based branch-and-bound method for its solution that significantly outperforms the direct solution of a nonanticipative formulation of the problem and, when combined with our proposed sampling approximation, is the first method that can provide statistical bounds that improve upon perfect information bounds for decision-dependent stochastic programs having continuous distribution. Preliminary computational results indicate the promise of the approach. Interesting directions for future work include studying the potential to speed up the method by using decomposition methods to solve the two-stage stochastic programs that have to be solved to obtain bounds and testing the method on different information structures.

Conflict of interest

The authors declare that they have no conflict of interest.

References

Artstein Z (1999) Gains and costs of information in stochastic programming. Annals of Operations Research 85:129–152

Artstein Z, Wets RJB (1993) Sensors and information in optimization under stochastic uncertainty. Mathematics of Operations Research 18(3):523–547

Birge JR, Louveaux F (1997) Introduction to stochastic programming. Springer, New York

Boland N, Dumitrescu I, Froyland G, Kalinowski T (2016) Minimum cardinality non-anticipativity constraint sets for multistage stochastic programming. Mathematical Programming 157:69–93

Carøe CC, Schultz R (1999) Dual decomposition in stochastic integer programming. Operations Research Letters 24(1-2):37–45

Colvin M, Maravelias C (2009) A branch and cut framework for multi-stage stochastic programming problems under endogenous uncertainty. Computer Aided Chemical Engineering 27:255–260

Colvin M, Maravelias C (2010) Modeling methods and a branch and cut algorithm for pharmaceutical clinical trial planning using stochastic programming. European Journal of Operational Research 203:205–215

Dupačová J (2006) Optimization under exogenous and endogenous uncertainty. In: Proceedings of MME06, pp 131–136

Freimer M, Linderoth J, Thomas D (2012) The impact of sampling methods on bias and variance in stochastic linear programs. Computational Optimization and Applications 51:51–75, DOI 10.1007/s10589-010-9322-x

Goel V, Grossmann IE (2004) A stochastic programming approach to planning of offshore gas field developments under uncertainty in reserves. Computers and Chemical Engineering 28(8):1409–1429

Goel V, Grossmann IE (2006) A class of stochastic programs with decision dependent uncertainty. Mathematical Programming 108:355–394, DOI 10.1007/s10107-006-0715-7

Hellemo L, Barton PE, Tomasgard A (2018) Decision-dependent probabilities in stochastic programs with recourse. Computational Management Science 15:369–395

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

Linderoth JT, Savelsbergh MWP (1999) A computational study of search strategies in mixed integer programming. INFORMS Journal on Computing 11:173–187

Mak WK, Morton DP, Wood RK (1999) Monte Carlo bounding techniques for determining solution quality in stochastic programs. Operations Research Letters 24:47–56

McKay MD, Beckman RJ, Conover WJ (1979) A comparison of three methods for selecting values of input variables in the analysis of output from a computer code. Technometrics 21(2):239–245, URL http://www.jstor.org/stable/1268522

Mercier L, Van Hentenryck P (2008) Amsaa: A multistep anticipatory algorithm for online stochastic combinatorial optimization, Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), vol 5015 LNCS. Springer, URL www.scopus.com, cited By :18

Norkin VI, Pflug GC, Ruszczyński A (1998) A branch and bound method for stochastic global optimization. Mathematical programming 83:425–450

Solak S, Clarke JPB, Johnson EL, Barnes ER (2010) Optimization of r&d project portfolios under endogenous uncertainty. European Journal of Operational Research 207(1):420–433, DOI https://doi.org/10.1016/j.ejor.2010.04.032, URL https://www.sciencedirect.com/science/article/pii/S0377221710003516

Tarhan B, Grossmann IE, Goel V (2009) Stochastic programming approach for the planning of offshore oil or gas field infrastructure under decision-dependent uncertainty. Industrial and Engineering Chemistry Research 48(6):3078–3097

Thain D, Tannenbaum T, Livny M (2005) Distributed computing in practice: The condor experience. Concurrency and Computation: Practice and Experience 17:323–356

Van Slyke R, Wets RJ (1969) L-shaped linear programs with applications to optimal control and stochastic programming. SIAM J Appl Math 17:638–663
Vayanos P, Kuhn D, Rustem B (2011) Decision rules for information discovery in multi-stage stochastic programming. In: 2011 50th IEEE Conference on Decision and Control and European Control Conference, pp 7368–7373, DOI 10.1109/CDC.2011.6161382
Xu WL, Nelson BL (2013) Empirical stochastic branch-and-bound for optimization via simulation. Iie Transactions 45(7):685–698

Appendix: Details of Branching Variable Selection

Here we present the formulas used for computing $\hat{\Delta}_j^-$ which is used in the computation of the score for probing decision j in Section 3.1. We describe this separately for the cases when internal sampling and external sampling is used.

External Sampling

The first method for choosing $\hat{\Delta}_j^-$ requires estimating $\mathbb{E}_{\boldsymbol{\eta}_S}[R(\boldsymbol{\eta}_S)|\boldsymbol{\eta}_j = 1]$ and $\mathbb{E}_{\boldsymbol{\eta}_S}[R(\boldsymbol{\eta}_S)|\boldsymbol{\eta}_j = 0]$. Recall that when using external sampling, $F(S)$ is estimated via the formula

$$\mathbf{F}_N(S) = N^{-1} \sum_{\eta_S \in \hat{\mathrm{H}}_S} |\Omega_N(\eta_S)| \mathbf{R}_N(\eta_S).$$

If we define $\hat{\mathrm{H}}_S^{jt} = \{\eta_S \in \hat{\mathrm{H}}_S : \eta_j = t\}$, then we estimate

$$\mathbb{E}_{\boldsymbol{\eta}_S}[R(\boldsymbol{\eta}_S)|\boldsymbol{\eta}_j = t] \approx (B^{jt})^{-1} \sum_{\eta_S \in \hat{\mathrm{H}}_S^{jt}} |\Omega_N(\eta_S)| \mathbf{R}_N(\eta_S)$$

for $t = 0, 1$, where $B^{jt} = \sum_{\eta_S \in \hat{\mathrm{H}}_S^{jt}} |\Omega_N(\eta_S)|$. Note that this estimate uses all the same values $\mathbf{R}_N(\eta_S)$ that are already computed when computing the estimate $\mathbf{F}_N(S)$.

The second method for choosing $\hat{\Delta}_j^-$ requires estimating the covariance of $\boldsymbol{\eta}_j$ and $R(\boldsymbol{\eta}_S)$. This is computed via the estimate:

$$N^{-1} \sum_{\eta_S \in \hat{\mathrm{H}}_S} |\Omega_N(\eta_S)|(\eta_j - m_j)(\mathbf{R}_N(\eta_S) - \mathbf{F}_N(S)),$$

where $m_j = N^{-1} \sum_{\eta_S \in \hat{\mathrm{H}}_S} |\Omega_N(\eta_S)| \eta_j$.

Internal Sampling

Recall that in this case $F(S)$ is estimated from (15) as

$$\bar{\mathbf{F}}_{N_1,N_2}(S) = N_1^{-1} \sum_{k \in [N_1]} \bar{\mathbf{R}}_{N_2}(\eta_S^k).$$

If we define $\Omega^{tj} = \{k \in [N_1] : \eta_j = t\}$, then we estimate

$$\mathbb{E}_{\boldsymbol{\eta}_S}[R(\boldsymbol{\eta}_S)|\boldsymbol{\eta}_j = t] \approx |\Omega^{tj}|^{-1} \sum_{k \in \Omega^{tj}} \bar{\mathbf{R}}_{N_2}(\eta_S^k)$$

for $t = 0, 1$. Once again, this estimate uses all the same values $\bar{\mathbf{R}}_{N_2}(\eta_S^k)$ that are already computed when computing the estimate $\bar{\mathbf{F}}_{N_1,N_2}(S)$.

The covariance of $\boldsymbol{\eta}_j$ and $R(\boldsymbol{\eta}_S)$ is estimated via the formula

$$\hat{\Delta}_j^- = {N_1}^{-1} \sum_{k\in[N_1]} (\eta_j^k - m_j)(\bar{\mathbf{R}}_{N_2}(\eta_S^k) - \bar{\mathbf{F}}_{N_1,N_2}(S)), \tag{25}$$

where $m_j = (N_1)_{-1} \sum_{k\in N_1} \eta_j^k$.

The estimates when the support of $\boldsymbol{\eta}$ or $\boldsymbol{\xi}$ are small are adapted similarly.

Table 9 shows an estimate of the perfect information bound (PI Bound) and the standard error of our estimate, as well as an estimate of the value of the best solution our methods found for the instance (Best LB), and the standard error of the estimate.

Table 9: Bound Information for Instances in Computational Studies

Name	**PI Bound**	**S.Err.**	**Best LB**	**S.Err.**
J20_1	9739.2	66.4	7919.0	59.5
J20_2	11465.6	72.1	9551.0	54.8
J20_3	13565.2	62.2	11454.0	61.3
J25_1	10404.7	68.3	8278.5	56.1
J25_2	12576.6	96.6	9756.4	48.1
J25_3	11389.8	74.2	8882.9	49.5
J20_1_C	11287.9	62.4	9278.4	29.1
J20_2_C	11631.4	52.1	9302.1	56.4
J20_3_C	12411.7	65.0	10015.2	39.8
J25_1_C	12330.1	49.3	9971.6	20.6
J25_2_C	10801.0	59.9	8787.1	31.8
J25_3_C	9796.7	49.3	7552.6	6.8