

Highlights

ROBIST: Robust Optimization by Iterative Scenario Sampling and Statistical Testing

Justin Starreveld, Guanyu Jin, Dick den Hertog, Roger J. A. Laeven

- Simple and effective data-driven algorithm for optimization under uncertainty
- Applicable to wide variety of stochastic optimization problems
- Numerical experiments demonstrate superior performance in comparison to alternative methods
- Python package publicly available, see <https://github.com/JustinStarreveld/ROBIST>

ROBIST: Robust Optimization by Iterative Scenario Sampling and Statistical Testing

Justin Starreveld^{a,*}, Guanyu Jin^b, Dick den Hertog^a, Roger J. A. Laeven^b

^a*Department of Business Analytics, University of Amsterdam, Plantage
Muidersgracht 12, Amsterdam, 1018 TV, North Holland, Netherlands*

^b*Department of Quantitative Economics, University of
Amsterdam, Roetersstraat 11, Amsterdam, 1018 WB, North Holland, Netherlands*

Abstract

In this paper, we propose *ROBIST*, a simple, yet effective, data-driven algorithm for optimization under parametric uncertainty. The algorithm first generates solutions in an iterative manner by sampling and optimizing over a relatively small set of scenarios. Then, using statistical testing, the robustness of the solutions is evaluated, which can be done with a much larger set of scenarios. ROBIST offers a number of practical advantages over existing methods as it is: (i) easy to implement, (ii) able to deal with a wide range of problems and (iii) capable of providing sharp probability guarantees that are easily computable and independent of the dimensions of the problem. Numerical experiments demonstrate the effectiveness of ROBIST in comparison to alternative methods.

Keywords: optimization, uncertainty, stochastic, robust, data-driven

2020 MSC: 90C15, 90C17

1. Introduction

The field of optimization under parametric uncertainty has undergone rapid development over the past few decades. However, despite this development, we observe that the existing methods in this field are still underutilized in practice. In this paper, we propose a new method that is able to circumvent some of the practical limitations of existing methods.

We propose an algorithm for treating uncertain convex programs (UCP), which appear in a wide variety of real-world problems such as supply chain planning, portfolio optimization, inventory control, engineering design, and so on. Such problems can be formulated as follows:

$$\begin{aligned} \max_{\mathbf{x} \in \mathcal{X}} \quad & g(\mathbf{x}) \\ \text{s.t.} \quad & f(\mathbf{x}, \mathbf{z}) \leq 0, \end{aligned} \tag{UCP}$$

where $\mathbf{x} \in \mathbb{R}^{d_x}$ is a decision vector, restricted to a closed convex feasible set \mathcal{X} , $\mathbf{z} \in \mathbb{R}^{d_z}$ is an uncertain parameter vector, $g(\mathbf{x})$ is a concave function and $f(\mathbf{x}, \mathbf{z})$ is a scalar-valued function that is convex in \mathbf{x} (for any \mathbf{z}). Without loss of generality, we can assume here that there is no uncertainty in the objective function $g(\mathbf{x})$, as one can always move

*Corresponding author

Email address: `j.s.starreveld@uva.nl` (Justin Starreveld)

the uncertainty to the constraints by using an epigraph formulation. Furthermore, note that multiple constraints $f_1(\mathbf{x}, \mathbf{z}) \leq 0, \dots, f_m(\mathbf{x}, \mathbf{z}) \leq 0$ can be incorporated into a single constraint by defining $f(\mathbf{x}, \mathbf{z}) := \max_{j=1, \dots, m} \{f_j(\mathbf{x}, \mathbf{z})\} \leq 0$.

Problem (UCP) as formulated above is not well-defined as it does not specify how the uncertain constraint $f(\mathbf{x}, \mathbf{z}) \leq 0$ should be treated. In the context of this paper, we assume that one is interested in obtaining a “robust” solution to (UCP), i.e., a solution \mathbf{x} that is “likely” to be feasible despite the uncertainty in regard to the parameter \mathbf{z} . One way to define this more precisely in mathematical terms, is to formulate (UCP) as a chance-constrained program (CCP):

$$\begin{aligned} \max_{\mathbf{x} \in \mathcal{X}} \quad & g(\mathbf{x}) \\ \text{s.t.} \quad & \mathbb{P}(f(\mathbf{x}, \tilde{\mathbf{z}}) \leq 0) \geq 1 - \epsilon, \end{aligned} \tag{1}$$

where the uncertain parameter \mathbf{z} is modeled as a random variable $\tilde{\mathbf{z}}$ with a probability distribution \mathbb{P} and ϵ represents some acceptable probability of constraint violation. First proposed by Charnes and Cooper (1959), such formulations have been widely studied within the field of stochastic programming (see e.g., Shapiro et al., 2009; Birge and Louveaux, 2011) and applied to a variety of problems (Birge, 1997; Wallace and Ziemba, 2005). However, the “true” probability distribution \mathbb{P} is often unknown in practice. Moreover, even if \mathbb{P} is known, exact tractable reformulations of (1) are only known for a limited number of situations (Shapiro and Nemirovski, 2005). As such, various data-driven techniques have been proposed to deal with the ambiguous nature of \mathbb{P} and alternative problem formulations have been proposed to circumvent probabilistic constraints such as (1).

1.1. Existing Approaches and Their Practical Limitations

In the following paragraphs we review four well-known approaches for dealing with uncertain constraints and highlight limitations to their application in practice. Though each approach formulates the problem differently, they share a common goal: to obtain a solution that satisfies Constraint (1) with a sufficient degree of confidence.

1.1.1. Sample Average Approximation.

The sample average approximation (SAA) approach replaces the unknown probability in (1) with an empirical distribution, constructed from a data set $\{\mathbf{z}^1, \dots, \mathbf{z}^N\}$. That is, Constraint (1) is replaced by:

$$\frac{1}{N} \sum_{i=1}^N \mathbb{1}_{\{f(\mathbf{x}, \mathbf{z}^i) \leq 0\}} \geq 1 - \epsilon, \tag{2}$$

where $\mathbb{1}_{\{f(\mathbf{x}, \mathbf{z}^i) \leq 0\}}$ indicates whether the constraint is satisfied for the realization \mathbf{z}^i . Theoretical results from Luedtke and Ahmed (2008) and Pagnoncelli et al. (2009) show that if the sample size N is sufficiently large, SAA offers a good approximation of (1), both in terms of optimal objective value and feasible region. However, the theoretically required sample size can be prohibitively large in practice. Moreover, Constraint (2) is typically non-convex. Such problems are commonly solved by reformulating the problem as a mixed-integer optimization problem (Luedtke et al., 2010; Luedtke, 2014), or by resorting to convex inner approximations, such as the CVaR method by Nemirovski and Shapiro (2007), or the ALSO-X and ALSO-X+ methods by Ahmed et al. (2017) and Jiang and Xie (2022).

1.1.2. Robust Optimization.

Robust optimization (RO) operates in a fully deterministic paradigm. In RO, one constructs an “uncertainty set” \mathcal{U} and enforces the constraint to hold for all realizations \mathbf{z} within the set \mathcal{U} . The uncertain constraint $f(\mathbf{x}, \mathbf{z}) \leq 0$ is thus formalized as:

$$f(\mathbf{x}, \mathbf{z}) \leq 0, \quad \forall \mathbf{z} \in \mathcal{U}. \quad (3)$$

This approach has emerged as a tractable alternative to stochastic programming for high-dimensional problems, see, for example, Bandi and Bertsimas (2012).

Depending on the distributional assumptions made regarding \mathbf{z} , Constraint (3) with a properly chosen uncertainty set \mathcal{U} can be used as a tractable reformulation of (1), or at least a safe approximation of it (see e.g., Bertsimas et al., 2018; Hong et al., 2021).

For an extended overview of RO and its applications, we refer to Ben-Tal et al. (2009) and Bertsimas and den Hertog (2022).

While RO has proven to be an effective approach in various applications, its effectiveness is highly dependent on the choice of the uncertainty set \mathcal{U} . As we demonstrate via numerical experiments in this paper, some of the aforementioned RO methods may scale poorly in the dimension of the uncertain parameter $d_{\mathbf{z}}$, or result in overly conservative solutions. The conservativeness of RO methods is a well known limitation of the “hard” robust constraint approach. However, we note that many alternative approaches have been proposed to alleviate this issue, see e.g., Fischetti and Monaci (2009), Ben-Tal et al. (2010, 2017) and Roos and den Hertog (2020).

In many cases, RO relies on the ability to reformulate (3) to a tractable robust counterpart, which is not always possible in practice. Moreover, certain reformulated robust counterpart may involve complex nonlinear constraints and/or an unacceptably large number of additional variables and constraints (Bertsimas et al., 2011). Furthermore, if the function f is non-concave in \mathbf{z} , exact reformulations of (3) are known only for specific combinations of the function f and uncertainty set \mathcal{U} (Bertsimas and den Hertog, 2022, Chapter 16). For the general case, safe approximations can be derived (Bertsimas et al., 2023). However, this requires many additional variables, which, along with the complexity of the methodology, may pose significant hindrances to its application in practice.

1.1.3. Distributionally Robust Optimization.

In distributionally robust optimization (DRO), the distribution \mathbb{P} is regarded as uncertain, yet restricted to an “ambiguity set” \mathcal{P} of possible distributions. The uncertain constraint can then be formulated as:

$$\mathbb{P}(f(\mathbf{x}, \tilde{\mathbf{z}}) \leq 0) \geq 1 - \epsilon, \quad \forall \mathbb{P} \in \mathcal{P}. \quad (4)$$

If \mathcal{P} contains the true probability distribution, then a solution that satisfies (4) also satisfies (1). For an overview of DRO we refer to Rahimian and Mehrotra (2019). For a survey on methods for dealing with ambiguous stochastic constraints such as (4) we refer to Postek et al. (2018).

Similarly to RO, the key challenge in DRO lies in the choice of the ambiguity set \mathcal{P} . Data-driven DRO has emerged as a popular approach, where one uses data to determine the ambiguity set \mathcal{P} . This can be done by estimation of the statistical moments of the distribution, see e.g., the method proposed by Delage and Ye (2010), or by using distance measures, see e.g., Mohajerin Esfahani and Kuhn (2018). Data-driven DRO offers advantages over RO in terms of conservativeness, however this may come at the cost of increased computational effort (Wang et al., 2024).

The ability to reformulate (4) to a tractable robust counterpart is, as with RO, dependent on the situation and not always possible in practice. There exist settings in which exact reformulations of (4) are possible, see, for example, Calafiore and Ghaoui (2006) and Jiang and Guan (2016). Various types of ambiguity sets with tractable counterparts are presented in Hanasusanto et al. (2015) and Postek et al. (2016). Nevertheless, DRO suffers from the same practical limitations as RO in terms of its general applicability and ease of implementation.

1.1.4. Scenario Optimization.

Scenario optimization (SO) is a data-driven technique that enforces the uncertain constraint to be satisfied for all elements in a set of scenario’s (see Calafiore and Campi, 2005; Nemirovski and Shapiro, 2006). More precisely, given a set of i.i.d. data $\{\mathbf{z}^1, \dots, \mathbf{z}^N\}$, the scenario optimization approach aims to solve the following scenario convex program (SCP):

$$\begin{aligned} \max_{\mathbf{x} \in \mathcal{X}} \quad & g(\mathbf{x}) \\ \text{s.t.} \quad & f(\mathbf{x}, \mathbf{z}^i) \leq 0, \quad \forall \mathbf{z}^i \in \mathcal{S}. \end{aligned} \tag{SCP}$$

The SO approach is straightforward to implement and can be applied to a wide variety of problems (e.g., it does not require concavity of f in \mathbf{z}). Furthermore, an elegant, theoretical result established by Calafiore and Campi (2005) and later tightened by Campi and Garatti (2008) connects the number of randomly sampled scenarios included in \mathcal{S} with the robustness of solutions obtained by solving (SCP). This result allows one to assert, with a certain level of confidence, that any solution to (SCP), where the scenarios in \mathcal{S} are randomly sampled from \mathbb{P} , satisfies probability guarantee (1). We refer to Theorem 1 in Campi and Garatti (2008) for the details.

The practical limitations of this approach are the following. First, while the result from Campi and Garatti (2008) was proven tight for the special class of “fully-supported” (UCP), this method can be overly conservative for various problems encountered in practice, as we demonstrate via numerical experiments in this paper. Second, the required number of randomly sampled scenarios grows linearly with the number of decision variables (Oishi, 2007). For large-scale problems, a large number of sampled scenarios implies a large number of dense constraints in (SCP), which can make solving the problem numerically challenging (Bertsimas et al., 2018). As such, the classic approach proposed by Calafiore and Campi (2005) is considered generally impractical for medium- and large-scale optimization problems.

A variety of methods have been proposed to remedy these limitations. For an overview on such methods, we refer to Alamo et al. (2015). In the numerical experiments presented in Section 4.3 of this paper, we apply the methods proposed by Carè et al. (2014); Calafiore (2017); Garatti et al. (2022) and demonstrate that these methods remain limited in their ability to deal with large-scale problems.

1.2. Our Method and its Advantages

Our method utilizes scenario optimization to generate solutions. However, instead of utilizing an *a priori* probabilistic guarantee, as in the classic approach of Calafiore and Campi (2005), we employ *a posteriori* probabilistic guarantees, which are derived via statistical testing. We do this because *a posteriori* guarantees, which are computed *after* the solution \mathbf{x} is known, are often significantly tighter than *a priori* guarantees, which are

derived *before* \mathbf{x} is known (Guzman et al., 2016; Shang and You, 2020; Bertsimas et al., 2021). This key difference allows our method to utilize a smaller set of scenarios when solving (SCP), which is computationally advantageous.

The use of a posteriori evaluations is not unique to our method. For example, this is also used in the works by Chamanbaz et al. (2015) and Calafiore (2017). However, our evaluation procedure for assessing the “robustness” of solutions differs from theirs in significant ways. First, our a posteriori testing procedure is based on a ϕ -divergence test statistic. It provides an asymptotic confidence lower bound on the true probability of feasibility of a solution according to the central limit theorem, requiring only an independence assumption. The tightness of the guarantee depends only on the size of the test. By contrast, the probabilistic guarantee provided by the bounds in Chamanbaz et al. (2015) and Calafiore (2017) requires, in addition to the independence assumption, that the optimization problems have unique optimal solutions or that a suitable tie-break rule is applied. Furthermore, the tightness of their guarantee depends on whether the optimization problem is “fully supported”. Second and importantly, the algorithm proposed in Chamanbaz et al. (2015) requires a new set of validation samples with increasing sample size at each iteration. Our a posteriori evaluation procedure does not require this, being unaffected by the number of iterations performed. Third, the application of our confidence bound is not only limited to chance-constrained problems, but can also be extended to optimization problems involving expectation-based risk measures.

Besides the aforementioned works, a “wait-and-judge” approach is also proposed in Campi and Garatti (2018) and applied in the method of Garatti et al. (2022). However, this evaluation procedure assesses the “robustness” of a solution by counting the number of “support constraints”, which also differs significantly from our statistical testing approach.

An important aspect of our approach is that the statistical tests are carried out on the *univariate* Bernoulli random variable $\mathbb{1}_{[f(\mathbf{x}, \tilde{\mathbf{z}}) \leq 0]}$, where \mathbf{x} is fixed. Since we are only concerned with the feasibility of \mathbf{x} , there are only two “classes” (i.e., $f(\mathbf{x}, \tilde{\mathbf{z}}) \leq 0$ and $f(\mathbf{x}, \tilde{\mathbf{z}}) > 0$). This provides probability guarantees that are independent of the dimension of the decision variables as well as the dimension of the uncertain parameters. This allows our evaluation procedure to scale better, as the problem size and/or the amount of data increases, than the evaluation procedures proposed by Yanıkoğlu and den Hertog (2013) and Campi and Garatti (2018), which is demonstrated in Sections 4.1.1 and 4.3.1.

Our main contribution is to propose a method that offers practical advantages over existing methods in the literature. The advantages of ROBIST are the following.

First, our method is computationally more efficient than many existing methods. By iteratively selecting which scenario(s) to sample and optimize for, we can reduce the total number of scenarios with which (SCP) is solved. Our evaluation procedure is independent of $d_{\mathbf{x}}$ and $d_{\mathbf{z}}$ and scales well with the amount of data available, which allows the algorithm to efficiently deal with large-scale optimization problems.

Second, our method is more versatile than many existing methods, as it is able to deal with a wide variety of problem types. These include optimization problems with: joint chance-constraints, constraints with non-concave uncertainty, expectation-based risk measures, regret-based risk measures and adaptive decision variables with non-fixed recourse.

In an effort to lower the hurdle for practical usage, we have implemented ROBIST in Python, which is a popular programming language amongst practitioners. The code is publicly available at: <https://github.com/JustinStarreveld/ROBIST>.

1.3. Structure

The remainder of the paper is organized as follows. In Section 2, we describe the ROBIST algorithm with the help of an illustrative example and provide theoretical analysis of its convergence. In Section 3, we discuss various possible generalizations and extensions. Then, in Section 4, we compare ROBIST to existing methods and demonstrate the efficiency and versatility of our proposed method via a variety of numerical experiments. Finally, we provide concluding remarks in Section 5. Additional technical details, proofs and extra numerical results are relegated to the Appendices.

1.3.1. Notation.

We denote a random variable by the tilde sign, i.e., \tilde{x} . Lowercase bold letters such as \mathbf{x} denote vectors, where \mathbf{e} denotes a vector of all ones. Calligraphic uppercase characters such as \mathcal{X} denote sets.

2. Methodology

As discussed in Section 1, explicitly modeling and optimizing over a constraint such as (1) is difficult. Moreover, the underlying probability distribution \mathbb{P} of $\tilde{\mathbf{z}}$ is rarely known in practice. However, given a data set $\mathcal{D}_N = \{\mathbf{z}^1, \dots, \mathbf{z}^N\}$ of N independent realizations of $\tilde{\mathbf{z}}$ and a solution \mathbf{x} , it is possible to use statistical testing to assert, with confidence greater than or equal to $1 - \alpha$, that \mathbf{x} satisfies the following probabilistic guarantee:

$$\mathbb{P}(f(\mathbf{x}, \tilde{\mathbf{z}}) \leq 0) \geq \gamma. \quad (5)$$

Henceforth, we will refer to γ as a *feasibility certificate*. Note that if a solution \mathbf{x} satisfies (5) with $\gamma \geq 1 - \epsilon$, one can state, with a certain level of confidence, that \mathbf{x} satisfies (1). This insight serves as the foundation to our proposed algorithm.

2.1. Algorithm

Our algorithm, ROBIST, consists of two main procedures: (i) a generation procedure in which we use a training data set $\mathcal{D}_{N_1}^{\text{train}} = \{\hat{\mathbf{z}}^1, \dots, \hat{\mathbf{z}}^{N_1}\}$ to generate solutions, and (ii) a two-step evaluation procedure in which we use a validation data set $\mathcal{D}_{N_2}^{\text{valid}} = \{\tilde{\mathbf{z}}^1, \dots, \tilde{\mathbf{z}}^{N_2}\}$ to evaluate all the candidate solutions generated by ROBIST, and a third test set $\mathcal{D}_{N_3}^{\text{test}} = \{\tilde{\mathbf{z}}^1, \dots, \tilde{\mathbf{z}}^{N_3}\}$ that evaluates the robustness of the final solution selected from the candidates. By embedding these two procedures in an iterative algorithm, we look to obtain solutions \mathbf{x} that satisfy (5) with $\gamma \geq 1 - \epsilon$, while minimizing the objective function $g(\mathbf{x})$. A complete description of ROBIST is provided using pseudo-code in Algorithm 1.

2.1.1. Generation procedure.

In each iteration i of the algorithm, we solve the following variant of (SCP) to generate solution \mathbf{x}_i :

$$\begin{aligned} \max_{\mathbf{x} \in \mathcal{X}} \quad & g(\mathbf{x}) \\ \text{s.t.} \quad & f(\mathbf{x}, \hat{\mathbf{z}}^j) \leq 0, \quad \forall \hat{\mathbf{z}}^j \in \mathcal{S}_i, \end{aligned} \quad (\text{SCP}_i)$$

where $\mathcal{S}_i \subseteq \mathcal{D}_{N_1}^{\text{train}}$ is a finite set of scenarios. The generated solution \mathbf{x}_i is required to be feasible for all scenarios $\hat{\mathbf{z}}^j$ in \mathcal{S}_i . Optimizing over a larger set \mathcal{S}_i is therefore likely to result in a more robust solution \mathbf{x}_i . However, to avoid overly conservative solutions and reduce the computational cost of solving (SCP_i) , our algorithm is designed to keep the size of \mathcal{S}_i to a minimum.

2.1.2. Evaluation procedure.

Given a data set $\mathcal{D}_N = \{\mathbf{z}^1, \dots, \mathbf{z}^N\}$ and a solution \mathbf{x}_i , one can evaluate the robustness of \mathbf{x}_i using \mathcal{D}_N and derive a feasibility certificate γ_i via statistical testing. Our procedure is as follows.

First, for each scenario $\mathbf{z}^j \in \mathcal{D}_N$, we compute $f(\mathbf{x}_i, \mathbf{z}^j)$ and check whether $f(\mathbf{x}_i, \mathbf{z}^j) \leq 0$ is satisfied. This provides an empirical estimate p of the probability that \mathbf{x}_i is feasible:

$$p := \frac{1}{N} \sum_{j=1}^N \mathbb{1}_{[f(\mathbf{x}_i, \mathbf{z}^j) \leq 0]}. \quad (6)$$

Second, we construct a statistical confidence interval around (6) using the modified χ^2 -distance, which is a member of the family of ϕ -divergences (see Appendix A for further details). This results in the following $1 - \alpha$ confidence region:

$$\mathcal{Q}_\phi(p, N, \alpha) := \left\{ q \in \mathbb{R} : q \geq 0, \frac{(q-p)^2}{p} + \frac{(q-p)^2}{1-p} \leq \frac{\chi_{1,1-\alpha}^2}{N} \right\}, \quad (7)$$

where α is determined by the user and $\chi_{1,1-\alpha}^2$ is the $1 - \alpha$ quantile of the chi-squared distribution with 1 degree of freedom. As shown by Pardo (2006), as $N \rightarrow \infty$, $\mathcal{Q}_\phi(p, N, \alpha)$ contains the true probability that \mathbf{x}_i is feasible with confidence of at least $1 - \alpha$. We note that this confidence region (7) is one-dimensional, which allows us to bypass the curse of dimensionality stemming from the number of the uncertain parameters $d_{\mathbf{z}}$ (in contrast to the ϕ -divergence-based confidence sets utilized in Yanıkoğlu and den Hertog, 2013; Ben-Tal et al., 2013). Furthermore, we note that (7) defines a family of possible distributions (i.e., an ambiguity set), which captures a degree of ambiguity (dependent on α) in the underlying probability distribution of $\mathbb{1}_{[f(\mathbf{x}_i, \mathbf{z}^j) \leq 0]}$.

Third, we determine feasibility certificate γ_i , which implies a probabilistic guarantee equivalent to (5) for solution \mathbf{x}_i , by computing:

$$\gamma_i := \min_{q \in \mathcal{Q}_\phi(p, N, \alpha)} q. \quad (8)$$

This can be easily computed, as shown by the following lemma (see Appendix B.1 for the proof).

Lemma 1. *The problem stated in (8) has the following closed-form solution:*

$$\min_{q \in \mathcal{Q}_\phi(p, N, \alpha)} q = \max \left\{ p - \sqrt{p(1-p)r}, 0 \right\}, \quad \text{with } r = \frac{\chi_{1,1-\alpha}^2}{N}. \quad (9)$$

Furthermore, γ_i is an increasing function in p .

Note that certificates derived via (8) are only statistically valid if the scenarios $\mathbf{z}^j \in \mathcal{D}_N$ are i.i.d. samples. It is for this reason that ROBIST separates the available data into training, validation and testing data sets. The purpose of the validation data $\mathcal{D}_{N_2}^{\text{valid}}$ is to derive a reasonable proxy certificate $\tilde{\gamma}_i$ for each generated solution \mathbf{x}_i , which we use to select a final solution. Then, to account for multiple hypothesis testing¹ and optimization

¹Though each feasibility certificate is valid for each solution, it does not hold simultaneously for all solutions. Hence, adjustment for multiple hypothesis testing is needed.

bias, the test data $\mathcal{D}_{N_3}^{\text{test}}$ is used to derive a statistically valid feasibility certificate for the final solution. Note that our probabilistic guarantee is only asymptotically valid (as $N_3 \rightarrow \infty$).

Furthermore, note that there are many different ways to derive a feasibility certificate γ_i via statistical testing. Our proposed evaluation procedure is primarily motivated by computational efficiency.

Remark 1. *Alternatively, one can also use the validation set $\mathcal{D}_{N_2}^{\text{valid}}$ to perform multiple hypothesis testing by constructing feasibility certificates that hold simultaneously for all candidate solutions. This can be done by enlarging the confidence region (7). However, as we observed in practice, this approach can be too conservative and may not be as effective as using a third test sample for the final assessment.*

Algorithm 1 ROBIST

Input: Sets $\mathcal{D}_{N_1}^{\text{train}} = \{\hat{\mathbf{z}}^1, \dots, \hat{\mathbf{z}}^{N_1}\}$, $\mathcal{D}_{N_2}^{\text{valid}} = \{\check{\mathbf{z}}^1, \dots, \check{\mathbf{z}}^{N_2}\}$ and $\mathcal{D}_{N_3}^{\text{test}} = \{\ddot{\mathbf{z}}^1, \dots, \ddot{\mathbf{z}}^{N_3}\}$.

Acceptable probability of constraint violation ϵ , statistical confidence level α , time limit T_{max} , iteration limit i_{max} and probability of taking opposite action v .

Output: Best found solution \mathbf{x}_{i^*} .

- 1: Let T represent the current running time of the algorithm
 - 2: Initialize set \mathcal{S}_0 with nominal or random scenario from $\mathcal{D}_{N_1}^{\text{train}}$
 - 3: Set iteration counter $i \leftarrow 0$
 - 4: **while** $T < T_{\text{max}}$ and $i < i_{\text{max}}$ **do**
 - 5: Solve (SCP _{i}) to obtain \mathbf{x}_i
 - 6: Derive proxy certificate $\hat{\gamma}_i$ via (9) using the scenarios in $\mathcal{D}_{N_1}^{\text{train}}$
 - 7: Draw random variable $\iota \sim U(0, 1)$
 - 8: **if** ($\hat{\gamma}_i \leq 1 - \epsilon$ **and** $\iota > v$) **or** ($\hat{\gamma}_i > 1 - \epsilon$ **and** $\iota < v$) **then**
 - 9: Randomly add a scenario from $\{\hat{\mathbf{z}}^j \in \mathcal{D}_{N_1}^{\text{train}} : f(\mathbf{x}_i, \hat{\mathbf{z}}^j) > 0\}$ to \mathcal{S}_{i+1}
 - 10: **else**
 - 11: Randomly remove a scenario from \mathcal{S}_i to create \mathcal{S}_{i+1}
 - 12: **end if**
 - 13: $i \leftarrow i + 1$
 - 14: **end while**
 - 15: Derive proxy certificates $\check{\gamma}_j, j = 0, \dots, i - 1$ via (9) using the scenarios in $\mathcal{D}_{N_2}^{\text{valid}}$
 - 16: **if** $\exists \check{\gamma}_j : \check{\gamma}_j \geq 1 - \epsilon$ **then**
 - 17: $i^* := \operatorname{argmax}_j \{g(\mathbf{x}_j) : \check{\gamma}_j \geq 1 - \epsilon\}$
 - 18: **else**
 - 19: $i^* := \operatorname{argmax}_j \{\check{\gamma}_j\}$
 - 20: **end if**
 - 21: Derive valid feasibility certificate $\check{\gamma}_{i^*}$ via (9) using the scenarios in $\mathcal{D}_{N_3}^{\text{test}}$
 - 22: Return \mathbf{x}_{i^*} with certificate $\check{\gamma}_{i^*}$
-

2.1.3. The scenario selection strategy.

The initial set \mathcal{S}_0 contains only a single scenario. If available, we utilize the nominal scenario, otherwise we pick a random scenario from $\mathcal{D}_{N_1}^{\text{train}}$. Then, at each iteration i , the set \mathcal{S}_{i+1} is constructed by either adding a scenario to \mathcal{S}_i , or removing a scenario from \mathcal{S}_i .

We inform this decision by constructing a *proxy* certificate $\hat{\gamma}_i$, derived using $\mathcal{D}_{N_1}^{\text{train}}$ instead of $\mathcal{D}_{N_2}^{\text{valid}}$. The idea is that $\hat{\gamma}_i$, while statistically invalid, can act as an estimate

of $\check{\gamma}_i$ and steer the algorithm's generation procedure. If $\hat{\gamma}_i < 1 - \epsilon$, one suspects that the solution \mathbf{x}_i is insufficiently robust, implying that we should add a scenario to \mathcal{S}_i . If $\hat{\gamma}_i \geq 1 - \epsilon$, the solution \mathbf{x}_i might be overly conservative, implying that we should remove a scenario from \mathcal{S}_i . The decision of whether to add or remove a scenario, is mainly driven by this proxy certificate $\hat{\gamma}_i$. However, with user-defined probability v , the opposite action is taken. This random component is added to ensure theoretical convergence of our algorithm, which we discuss in Section 2.3. We analyze the effect of v using numerical experiments in Appendix C.1.1.

In an effort to keep the algorithm as simple as possible, when adding a scenario, we randomly pick a scenario from the set of currently violated scenarios $\{\hat{\mathbf{z}}^j \in \mathcal{D}_{N_1}^{\text{train}} : f(\mathbf{x}_i, \hat{\mathbf{z}}^j) > 0\}$. We numerically compare this strategy with alternative selection strategies in Appendix C.1.2. When removing a scenario, we randomly pick a scenario from \mathcal{S}_i .

We note that the efficiency of Algorithm 1 can, in certain cases, be improved. Whenever a scenario $\hat{\mathbf{z}}^j$ is removed from \mathcal{S}_i and the dual variable corresponding to the constraint $f(\mathbf{x}_i, \hat{\mathbf{z}}^j) \leq 0$ is zero, one can skip Step 5 as $\mathbf{x}_{i+1} = \mathbf{x}_i$. Furthermore, the evaluation of \mathbf{x}_{i+1} can be skipped in Steps 6 and 15, as $\hat{\gamma}_{i+1} = \hat{\gamma}_i$ and $\check{\gamma}_{i+1} = \check{\gamma}_i$.

Finally, we note that the proposed algorithm is highly parallelizable. First, the evaluations performed in Steps 6, 15 and 21 can be done independently per scenario. Second, one could conduct multiple while loops (lines 4-14 of Algorithm 1) in parallel. Doing so would result in less correlated sets \mathcal{S}_i and is likely to generate a more diverse set of solutions.

2.1.4. Stopping criteria and final solution.

The algorithm terminates when a prescribed time limit or a maximum number of iterations is reached. If the algorithm is not able to find a solution with feasibility certificate $\gamma_i \geq 1 - \epsilon$, it returns the solution with the highest certificate. Otherwise, it returns the solution \mathbf{x}_i with maximal objective value $g(\mathbf{x}_i)$, while requiring that $\gamma_i \geq 1 - \epsilon$.

2.2. Illustrative Example

Consider the following toy problem:

$$\max_{x_1, x_2 \leq 1} x_1 + x_2 \quad (10)$$

$$\text{s.t. } z_1 x_1 + z_2 x_2 \leq 1, \quad (11)$$

where z_1 and z_2 are uncertain parameters, both uniformly distributed with support $[-1, 1]$. Note that Problem (10)-(11) is an example of (UCP), where $\mathcal{X} = \{\mathbf{x} : x_1 \leq 1, x_2 \leq 1\}$, $g(\mathbf{x}) = x_1 + x_2$ and $f(\mathbf{x}, \mathbf{z}) = z_1 x_1 + z_2 x_2 - 1$.

Imagine we have access to a data set of $N = 300$ realizations of $(\tilde{z}_1, \tilde{z}_2)$ and would like a solution to satisfy Constraint (11) with probability greater than or equal to 90% (i.e., $\epsilon = 0.1$). In the following paragraphs we illustrate the application of Algorithm 1 to this problem.

First, we randomly split the data set into three equal-sized sets $\mathcal{D}_{N_1}^{\text{train}} = \{\hat{\mathbf{z}}^1, \dots, \hat{\mathbf{z}}^{N_1}\}$, $\mathcal{D}_{N_2}^{\text{valid}} = \{\check{\mathbf{z}}^1, \dots, \check{\mathbf{z}}^{N_2}\}$ and $\mathcal{D}_{N_3}^{\text{test}} = \{\ddot{\mathbf{z}}^1, \dots, \ddot{\mathbf{z}}^{N_3}\}$, each containing $N_1 = N_2 = N_3 = 100$ scenarios. We use $\mathcal{D}_{N_1}^{\text{train}}$ to generate and (informally) evaluate the robustness of solutions during the iterative phase of the algorithm. Then, in the evaluation phase of the algorithm, we utilize $\mathcal{D}_{N_2}^{\text{valid}}$ to evaluate the generated solutions and select the most promising solution. Finally, we utilize $\mathcal{D}_{N_3}^{\text{test}}$ to derive a statistically valid probability guarantee for the selected solution.

Suppose we initialize $\mathcal{S}_0 = \{\bar{\mathbf{z}}\}$ with the expected/nominal case $\bar{\mathbf{z}} = (z_1, z_2) = (0, 0)$. Then, solving (SCP_{*i*}) with \mathcal{S}_0 provides an initial solution: $\mathbf{x}_0 = (x_1, x_2) = (1, 1)$ with objective value $g(\mathbf{x}_0) = 2$. The next step is to use our evaluation procedure to assess the robustness of \mathbf{x}_0 . First we compute an empirical estimate of the probability of violating Constraint (11) by determining whether $f(\hat{\mathbf{z}}^j, \mathbf{x}_0) \leq 0, \forall \hat{\mathbf{z}}^j \in \mathcal{D}_{N_1}^{\text{train}}$. See Figure 1 for a visual aid.

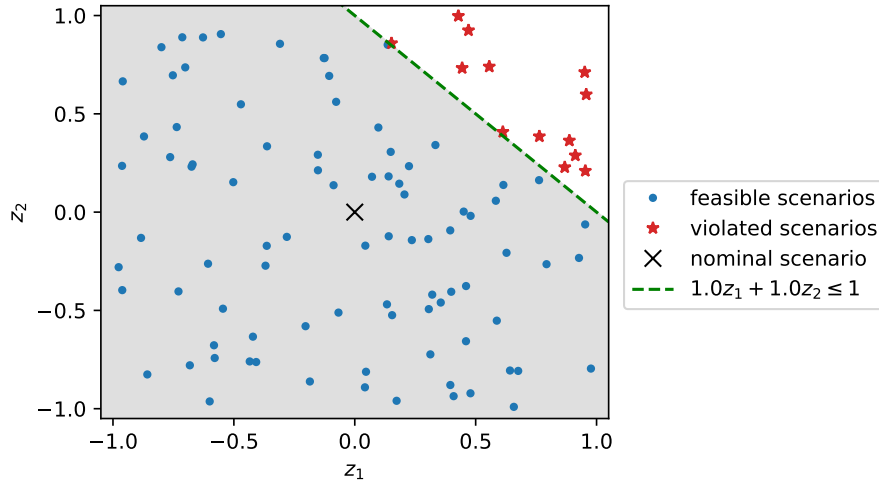


Figure 1: Visualization of \mathcal{S}_0 and the evaluation of the constraint $f(\mathbf{x}_0, \hat{\mathbf{z}}^j) \leq 0$ of the solution $\mathbf{x}_0 = (1, 1)$ on the training data $\mathcal{D}_{N_1}^{\text{train}}$. The data points for which the constraint is feasible/infeasible are indicated in blue/red.

We find that solution \mathbf{x}_0 is feasible for $\frac{87}{100}$ of the scenarios in the training data $\mathcal{D}_{N_1}^{\text{train}}$. Setting the probability of making a type I error less than or equal to 1% (i.e., $\alpha = 0.01$), we use Equation (9), with $p = 0.87$, $N = 100$ and $\alpha = 0.01$, to derive a proxy certificate of $\hat{\gamma}_0 = 0.78$.

Assume that we have set the probability of taking the opposite action $v = 0$. Then, as our proxy certificate $\hat{\gamma}_0$ does not yet meet the desired level of robustness ($\hat{\gamma}_0 = 0.78 < 1 - \epsilon = 0.90$) the algorithm will randomly pick one of the 13 currently violated scenarios (indicated by red stars in Figure 1) and add this scenario to our set. Suppose scenario $\hat{\mathbf{z}}^{11} = (0.96, 0.60)$ is chosen, then $\mathcal{S}_1 = \{\bar{\mathbf{z}}, \hat{\mathbf{z}}^{11}\}$ and we proceed to the next iteration.

Using an enlarged set of scenarios \mathcal{S}_1 , we solve (SCP_{*i*}) and retrieve solution: $\mathbf{x}_1 = (0.4, 1)$ with objective value $g(\mathbf{x}_1) = 1.4$. While adding a scenario/constraint to our optimization problem has lowered the objective value, it is likely to ensure that the resulting solution is more robust. Again, we evaluate the robustness of our newly generated solution \mathbf{x}_1 using the scenarios in $\mathcal{D}_{N_1}^{\text{train}}$ (see Figure 2). We find that our new solution \mathbf{x}_1 is feasible for $\frac{97}{100}$ of the scenarios, from which we derive a proxy certificate $\hat{\gamma}_1 = 0.93$.

As this exceeds our desired level of feasibility ($\hat{\gamma}_1 \geq 1 - \epsilon = 0.90$), the algorithm will remove a scenario from \mathcal{S}_1 in the following iteration. The algorithm continues adding or removing scenarios and evaluating the resulting solutions on $\mathcal{D}_{N_1}^{\text{train}}$ in this manner until either the time limit or iteration limit is reached.

In this example, we set a limit of 100 iterations and once this stopping criteria is reached, we use the “out-of-sample” validation data $\mathcal{D}_{N_2}^{\text{valid}}$ to evaluate each generated solution \mathbf{x}_i and obtain proxy feasibility certificates $\check{\gamma}_i$. These evaluations can then be used to construct a trade-off curve and aid in choosing the most promising solution. The orange line in Figure 3 depicts such a trade-off curve, where each orange square represents a non-dominated solution (with respect to the validation data).

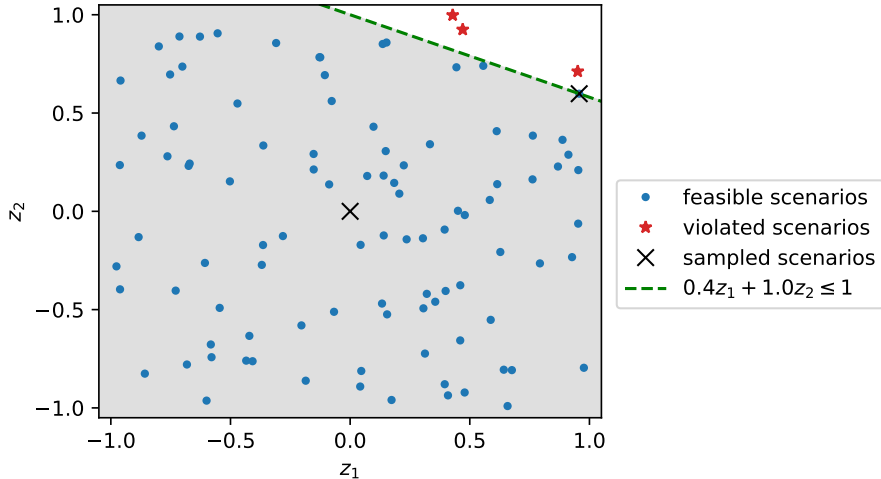


Figure 2: Visualization of \mathcal{S}_1 and the evaluation of solution $\mathbf{x}_1 = (0.4, 1)$ on the training data.

Recall that we would like our solution to be feasible with probability $\geq 90\%$, thus we select the solution with the highest objective value while requiring that the proxy certificate derived from the validation data is greater than or equal to 0.90. Finally, we utilize $\mathcal{D}_{N_3}^{\text{test}}$ to provide a valid feasibility certificate for this solution. In this example, the algorithm returns a solution to Problem (10)-(11) with objective value 1.37 and feasibility certificate 0.93.

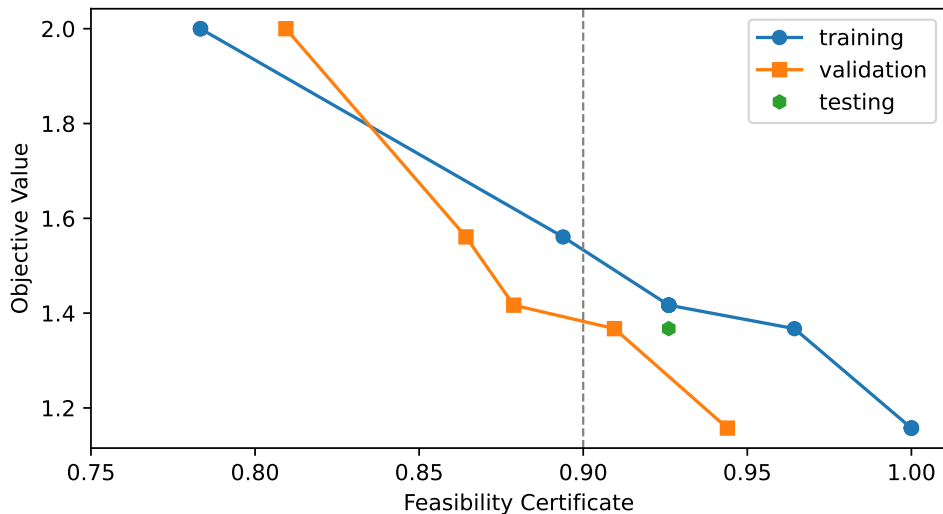


Figure 3: Trade-off curves constructed from a set of solutions generated by ROBIST. The blue circles depict proxy certificates derived using the training data, the orange squares represent proxy certificates derived using the validation data and the green hexagon represents the final feasibility certificate provided by the testing data. The vertical dotted line represents the desired level of robustness $(1 - \epsilon)$.

2.3. Convergence

In this section, we study the optimality of the final solution generated by Algorithm 1, which we define given a realization of the training and validation data sets ($\mathcal{D}_{N_1}^{\text{train}}$ and $\mathcal{D}_{N_2}^{\text{valid}}$). Let $\tilde{\mathcal{S}}_1, \dots, \tilde{\mathcal{S}}_{2^{N_1}}$ denote all the possible subsets of $\mathcal{D}_{N_1}^{\text{train}}$. Let $\mathbf{x}_i \in \arg\max_{\mathbf{x} \in \mathcal{X}} \{g(\mathbf{x}) \mid f(\mathbf{x}, \mathbf{z}^j) \leq 0, \forall \mathbf{z}^j \in \tilde{\mathcal{S}}_i\}$. Let $\tilde{\gamma}_i$ denote the proxy feasibility certificate of \mathbf{x}_i , as derived via (8) using $\mathcal{D}_{N_2}^{\text{valid}}$. Then, we have the following definition of optimality.

Definition 1. Given $\mathcal{D}_{N_1}^{\text{train}}$ and $\mathcal{D}_{N_2}^{\text{valid}}$, an optimal solution \mathbf{x}^* with respect to the data is defined as an element of the set $\text{argmax}_{i=1, \dots, 2N_1} \{g(\mathbf{x}_i) \mid \check{\gamma}_i \geq 1 - \epsilon\}$.

The following lemma shows that Algorithm 1 yields a solution that is optimal with respect to the training and validation data, if the modeler sets no limitation on the running time and the number of iterations of the algorithm (see Appendix B.2 for the proof).

Lemma 2. Assume that in Algorithm 1 the modeler has set $T_{\max} = i_{\max} = \infty$ and $v > 0$. Suppose that for all possible subsets $\mathcal{S}_i \subseteq \mathcal{D}_{N_1}^{\text{train}}$, the solution to (SCP_i) is unique, and that there exists a (SCP_i) such that its solution achieves a proxy certificate $\check{\gamma}_i \geq 1 - \epsilon$. Then, the probability that the solution obtained from Algorithm 1 after finitely many iterations, is optimal with respect to the data, is 1.

Remark 2. The assumption used in the proof of Lemma 2 on the uniqueness of the solutions of (SCP_i) for all possible subsets $\mathcal{S}_i \subseteq \mathcal{D}_{N_1}^{\text{train}}$ is unnecessary if the solver that is used to solve (SCP_i) does not output different solutions when a redundant constraint is removed or added. Here, we consider a constraint “redundant” if its addition or removal does not change the optimal objective value of the optimization problem.

3. Generalizations and Extensions

In this section we describe how ROBIST can be applied to a variety of problem types. Depending on the type of problem, certain modifications are made to Algorithm 1.

3.1. Uncertainty in the Objective Function

Although Algorithm 1 as described in Section 2 is already able to deal with parametric uncertainty in the objective function, minor modifications can be made to improve its performance. Consider an uncertain convex problem of the form:

$$\max_{\mathbf{x} \in \mathcal{X}} f(\mathbf{x}, \mathbf{z}). \quad (12)$$

Problem (12) can be reformulated such that the uncertain parameters only appear in the constraints of the problem, by introducing an epigraph variable θ :

$$\max_{\mathbf{x}, \theta} \theta \quad (13)$$

$$\text{s.t. } \theta - f(\mathbf{x}, \mathbf{z}) \leq 0, \quad (14)$$

$$\mathbf{x} \in \mathcal{X}. \quad (15)$$

Note that the epigraph variable $\theta \in \mathbb{R}$ can always be adjusted such that (14) is satisfied. As such, Constraint (14) should be treated differently.

We slightly alter Steps 15 and 21 in Algorithm 1 in the following manner. For a given solution (\mathbf{x}_i, θ_i) , instead of computing the value of γ_i for which we can claim with confidence of at least $1 - \alpha$ that:

$$\mathbb{P}(f(\mathbf{x}_i, \tilde{\mathbf{z}}) \geq \theta_i) \geq \gamma_i, \quad (16)$$

we are now interested in determining the maximum value of θ_i , let this be denoted by θ_i^* , for which we can claim, with confidence of at least $1 - \alpha$, that:

$$\mathbb{P}(f(\mathbf{x}_i, \tilde{\mathbf{z}}) \geq \theta_i^*) \geq 1 - \alpha. \quad (17)$$

We determine θ_i^* in the following manner. First, we determine (via golden-section search) the thresholds for the number of scenarios in $\mathcal{D}_{N_2}^{\text{valid}}$ and $\mathcal{D}_{N_3}^{\text{test}}$ for which (14) would have to be satisfied in order to claim (17). Denote these thresholds by N_2^{min} and N_3^{min} . Then, depending on whether we are in Step 15 or Step 21, we sort the function evaluations $f(\mathbf{x}_i, \mathbf{z}^j)$ for all scenarios $\mathbf{z}^j \in \mathcal{D}_{N_2}^{\text{valid}}$ (or $\mathcal{D}_{N_3}^{\text{test}}$) and set θ_i^* equal to the N_2^{min} -th (or the N_3^{min} -th) largest evaluation.

Finally, since there are no uncertain constraints in Problem (12), there is no trade-off between feasibility and optimality. In such situations, the problem becomes a one-dimensional search for the maximal θ_i^* for which we can claim (17). As such, Steps 16-20 in Algorithm 1 are also adjusted, where we now define $i^* := \operatorname{argmax}_j \{\theta_j^*\}$.

3.2. Adaptive Optimization Problems

In this subsection, we show how ROBIST can be extended to deal with two-stage adaptive optimization problems. We note that the approach can also be applied to multi-stage problems (with minor modifications). However, for ease of exposition, in this paper we focus on a two-stage setting. Consider the following problem:

$$\max_{\mathbf{x} \in \mathcal{X}} g(\mathbf{x}) \quad (18)$$

$$\text{s.t. } V(\mathbf{x}, \mathbf{z}) \geq 0, \quad (19)$$

where

$$V(\mathbf{x}, \mathbf{z}) := \max_{\mathbf{y} \in \mathcal{Y}(\mathbf{x})} f(\mathbf{x}, \mathbf{z}, \mathbf{y}). \quad (20)$$

Here the decision vector \mathbf{x} represents first-stage “here-and-now” decisions and the decision vector \mathbf{y} consists of second-stage “wait-and-see” decisions. The \mathbf{y} variables are adaptive, i.e., they are able to adapt to the realization of $\tilde{\mathbf{z}}$. Moreover, the second-stage decisions \mathbf{y} are restricted to some closed convex feasible set $\mathcal{Y}(\mathbf{x})$, which may depend on the first-stage decisions \mathbf{x} .

We are still able to generate solutions to Problem (18)-(20) by solving, at each iteration i , a scenario convex program with respect to some set of scenarios \mathcal{S}_i . However, this now involves the inclusion of *recourse* decision vectors \mathbf{y}^j for each scenario $\hat{\mathbf{z}}^j \in \mathcal{S}_i$. This amounts to solving the following optimization problem:

$$\max_{\mathbf{x}, \mathbf{y}} g(\mathbf{x}) \quad (21)$$

$$\text{s.t. } f(\mathbf{x}, \hat{\mathbf{z}}^j, \mathbf{y}^j) \geq 0, \quad \forall \hat{\mathbf{z}}^j \in \mathcal{S}_i, \quad (22)$$

$$\mathbf{y}^j \in \mathcal{Y}(\mathbf{x}), \quad \forall \hat{\mathbf{z}}^j \in \mathcal{S}_i, \quad (23)$$

$$\mathbf{x} \in \mathcal{X}. \quad (24)$$

By solving (21)-(24) we are able to generate a here-and-now solution \mathbf{x}_i at each iteration i . The evaluation of the solution requires more computation than in the static setting. Given a data set of N independent scenarios $\{\mathbf{z}^1, \dots, \mathbf{z}^N\}$, instead of performing simple function evaluations (e.g., evaluating $f(\mathbf{x}_i, \mathbf{z}^j)$ for $j = 1, \dots, N$), one now evaluates $V(\mathbf{x}_i, \mathbf{z}^j)$ for $j = 1, \dots, N$ by solving N instances of Problem (20). This allows one to determine whether there exists a recourse decision \mathbf{y} such that uncertain constraint (19) could be satisfied. With this information one can compute empirical estimate p , where:

$$p := \frac{1}{N} \sum_{j=1}^N \mathbf{1}_{[V(\mathbf{x}_i, \mathbf{z}^j) \leq 0]}.$$

This estimate p can then be used to derive feasibility certificate γ_i by computing (8).

3.3. Statistical Confidence Bounds on Expectation

Throughout this paper we primarily focus on obtaining a solution that is robust in the sense of a probability guarantee such as (5). However, our method can be extended to incorporate expectation-based risk measures. In this subsection, we discuss how one can use a posteriori statistical testing to derive (asymptotic) upper and lower confidence bounds on:

$$\mathbb{E}_{\mathbb{P}}[f(\mathbf{x}, \tilde{\mathbf{z}})], \quad (25)$$

where $\mathbb{E}_{\mathbb{P}}$ denotes the expectation with respect to \mathbb{P} , the distribution of $\tilde{\mathbf{z}}$.

Given a solution \mathbf{x}_i , assume that we have bounded support $[l_{\mathbf{x}_i}, u_{\mathbf{x}_i}]$ for $f(\mathbf{x}_i, \tilde{\mathbf{z}})$, and construct a partition $\{[e_k, e_{k+1}]\}_{k=1}^K$, where $l_{\mathbf{x}_i} = e_1 \leq e_2 \leq \dots \leq e_{K+1} = u_{\mathbf{x}_i}$. Then, the following inequality must hold:

$$\mathbb{E}_{\mathbb{P}}[f(\mathbf{x}_i, \tilde{\mathbf{z}})] = \sum_{k=1}^K \int_{\Omega} f(\mathbf{x}_i, \tilde{\mathbf{z}}) \mathbb{1}_{\{f(\mathbf{x}_i, \tilde{\mathbf{z}}) \in [e_k, e_{k+1}]\}} d\mathbb{P} \leq \sum_{k=1}^K e_{k+1} \mathbb{P}(f(\mathbf{x}_i, \tilde{\mathbf{z}}) \in [e_k, e_{k+1}]).$$

Let vector $\mathbf{p} \in \mathbb{R}^K$ be an empirical estimate of the probability (under \mathbb{P}) that $f(\mathbf{x}_i, \tilde{\mathbf{z}})$ resides in each of the K intervals. One can compute this estimate using N independent scenarios \mathbf{z}^j , where the k -th element of \mathbf{p} , p_k , is computed as follows:

$$p_k := \frac{1}{N} \sum_{j=1}^N \mathbb{1}_{[f(\mathbf{x}_i, \mathbf{z}^j) \in [e_k, e_{k+1}]]}.$$

Then, one can construct the following ϕ -divergence based $(1 - \alpha)$ -confidence region around \mathbf{p} (see Appendix A for further details):

$$\mathcal{Q}_{\phi}(\mathbf{p}, N, \alpha) = \left\{ \mathbf{q} \in \mathbb{R}^K : \mathbf{q} \geq \mathbf{0}, \mathbf{e}^{\top} \mathbf{q} = 1, I_{\phi}(\mathbf{q}, \mathbf{p}) \leq \frac{\phi''(1)}{2N} \chi_{K-1, 1-\alpha}^2 \right\}. \quad (26)$$

As $N \rightarrow \infty$, the set (26) contains the true probability that $f(\mathbf{x}_i, \tilde{\mathbf{z}})$ resides in each of the K intervals. As such, one can get an asymptotic upper confidence bound u_i on $\mathbb{E}_{\mathbb{P}}[f(\mathbf{x}_i, \tilde{\mathbf{z}})]$, by computing:

$$u_i := \max_{\mathbf{q} \in \mathcal{Q}_{\phi}(\mathbf{p}, N, \alpha)} \sum_{k=1}^K e_{k+1} q_k. \quad (27)$$

Similarly, one can also construct a lower confidence bound l_i on $\mathbb{E}_{\mathbb{P}}[f(\mathbf{x}_i, \tilde{\mathbf{z}})]$ by computing:

$$l_i := \min_{\mathbf{q} \in \mathcal{Q}_{\phi}(\mathbf{p}, N, \alpha)} \sum_{k=1}^K e_k q_k. \quad (28)$$

The optimization problems stated in (27) and (28) are both convex and easy to solve.

3.4. Regret-Based Guarantees

Within optimization under uncertainty it is also common to consider regret minimization. In this subsection, we show how our methodology can be extended to provide statistical guarantees in regard to the regret associated with any given solution \mathbf{x}_i .

Given a solution \mathbf{x}_i , we define the regret $R(\mathbf{x}_i, \mathbf{z}^j)$ with respect to a realized scenario \mathbf{z}^j , as:

$$R(\mathbf{x}_i, \mathbf{z}^j) := \begin{cases} g^*(\mathbf{z}^j) - g(\mathbf{x}_i), & \text{if } f(\mathbf{x}_i, \mathbf{z}^j) \leq 0, \\ +\infty, & \text{otherwise,} \end{cases} \quad (29)$$

where:

$$g^*(\mathbf{z}^j) := \max_{\mathbf{x} \in \mathcal{X}} \{g(\mathbf{x}) : f(\mathbf{x}, \mathbf{z}^j) \leq 0\}. \quad (30)$$

The regret measures the ex-post difference between the achieved objective value and the best objective value that could have been obtained if the realization of $\tilde{\mathbf{z}}$ had been known before making the decision.

Incorporating regret into our approach requires only a minor adjustment to the evaluation procedure. For each scenario $\mathbf{z}^j \in \mathcal{D}_N$, instead of evaluating $f(\mathbf{x}_i, \mathbf{z}^j)$, we evaluate $R(\mathbf{x}_i, \mathbf{z}^j)$, as defined in (29). These evaluations can then be used to claim, with confidence of at least $1 - \alpha$, that:

$$\mathbb{P}(R(\mathbf{x}_i, \tilde{\mathbf{z}}) \leq \tau) \geq \beta_i, \quad (31)$$

where τ represents some threshold value and β_i is an asymptotic lower bound on the probability that the regret of solution \mathbf{x}_i is less than τ . Additionally, using the approach described in Section 3.3, it is also possible to derive a $(1 - \alpha)$ -statistical confidence interval $[l_i, u_i]$ for the expected regret $\mathbb{E}_{\mathbb{P}}[R(\mathbf{x}_i, \tilde{\mathbf{z}})]$ for any given solution \mathbf{x}_i .

4. Numerical Experiments

In this section, we present numerical experiments to test the performance of ROBIST on five different applications:

1. Toy problem: comparison with Yanıkođlu and den Hertog (2013), with additional analysis of ROBIST (see Section 4.1 and Appendix C.1).
2. Linear CCP: comparison with SAA-based methods of Ahmed et al. (2017) and Jiang and Xie (2022) (see Section 4.2).
3. Weighted distribution problem: comparison with the scenario optimization methods of Calafiore and Campi (2005); Carè et al. (2014); Calafiore (2017) and Garatti et al. (2022) (see Section 4.3).
4. Portfolio management problem: comparison with the classical robust optimization approach of Bertsimas et al. (2018) (see Appendix C.2).
5. Two-stage adaptive lot-sizing problem: comparison with sampling approach of Vayanos et al. (2012) (see Appendix C.3).

For all experiments we utilize synthetic randomly generated data and split the data equally and randomly into the training, validation and testing data sets. Furthermore, in Step 2 of Algorithm 1 we initialize \mathcal{S}_0 with a random scenario from $\mathcal{D}_{N_1}^{\text{train}}$ and set $v = 0.01$ (see Appendix C.1.1 for numerical experiments that analyze the effect of this parameter).

All computations are conducted on a 64-bit Windows machine equipped with a 2.80 GHz Intel Core i7 processor with 32 GB of RAM. All mathematical programs are coded in Python 3.10 using CVXPY 1.3 and solved with Gurobi 10.0.0. The code is publicly available at <https://github.com/JustinStarreveld/ROBIST>.

4.1. Toy Problem

In this subsection, we consider the toy problem from Yanikoğlu and den Hertog (2013), which is similar to the illustrative example presented in Section 2.2, but now in k dimensions. The problem is formulated as follows:

$$\max_{\mathbf{x}} \mathbf{e}^\top \mathbf{x} \quad (32)$$

$$\text{s.t. } \mathbb{P}(\tilde{\mathbf{z}}^\top \mathbf{x} \leq 1) \geq 1 - \epsilon, \quad (33)$$

$$\mathbf{x} \leq 1, \quad (34)$$

where $\mathbf{x}, \mathbf{z} \in \mathbb{R}^k$, and $\mathbf{e}^\top = (1, 1, \dots, 1) \in \mathbb{R}^k$. The random variables $\tilde{z}_1, \dots, \tilde{z}_k$ are assumed to be independently and uniformly distributed in $[-1, 1]$.

4.1.1. Comparison with Yanikoğlu and den Hertog (2013).

In the following experiments, we compare ROBIST with the method proposed by Yanikoğlu and den Hertog (2013), which is abbreviated as Y&dH. For the numerical experiments presented in this subsection, we set $\epsilon = 0.05$, $\alpha = 0.01$ and $i_{\max} = 200$.

For the details in regards to Y&dH, we refer to Section 3.2 of their paper. In these experiments we copy their settings, using the modified χ^2 -distance as the ϕ -divergence function, along with an ellipsoidal uncertainty set with an initial radius of 0.1 and a step size of 0.01. Furthermore, we follow the implementation of Y&dH and construct the cells such that the support of $\tilde{\mathbf{z}}$ is divided into 10^k cells of equal geometry.

Two limitations mentioned in Yanikoğlu and den Hertog (2013) are that, as the number of uncertain parameters increases, (i) the required number of data points increases and (ii) the computational performance deteriorates. This is due to the evaluation procedure with which their probability guarantees are derived, which is dependent on the dimension of $\tilde{\mathbf{z}}$. By contrast, the probability guarantees utilized in ROBIST are independent of the dimension of $\tilde{\mathbf{z}}$. We illustrate this difference in the following numerical experiments.

For Y&dH, the dimension of $\tilde{\mathbf{z}}$ influences the number of “cells” and thus the amount of data required.² For ROBIST there is no strict minimum or maximum regarding the amount of data and the algorithm is given access to $N = 1000$ randomly generated realizations of $\tilde{\mathbf{z}}$.

In Table 1 we report the total computation time for both methods, as well as the best objective value (belonging to a solution for which the associated feasibility certificate is greater than or equal to $1 - \epsilon = 0.95$). To control for the effect of randomness, the experiment is repeated 100 times and we report the average.

Even for the relatively small problem instances considered in Table 1, we find that, as the problem size k increases, the amount of data required by Y&dH quickly becomes unmanageable and the computational performance of the method deteriorates. In contrast, for ROBIST we observe that having access to 1,000 data points is sufficient for these problem instances and that the increase in time is relatively small.

We also observe that ROBIST is able to provide solutions with higher objective values than the solutions generated by Y&dH, while possessing comparable feasibility certificates. Even though the target probability of constraint satisfaction was set to 0.95, we find,

²We adhere to the rule of thumb stated in Yanikoğlu and den Hertog (2013) that each cell should contain “at least five observations”. It follows that, when applying Y&dH to this problem, a minimum of 5×10^k data points is required. To be on the safe side, Y&dH is provided with twice this minimum amount (i.e., 10^{k+1} randomly generated data points).

Table 1: Results from applying Yanıkoğlu and den Hertog (2013) and ROBIST to Problem (32)-(34) as the dimension of the problem (represented by k) increases. Here, N indicates the total amount of data utilized by the respective methods.

| k | N | | Time (s) | | Objective value | | Feasibility certificate | |
|-----|---------|--------|----------|--------|-----------------|--------|-------------------------|--------|
| | Y&dH | ROBIST | Y&dH | ROBIST | Y&dH | ROBIST | Y&dH | ROBIST |
| 2 | 1000 | 1000 | 4.3 | 1.4 | 1.19 | 1.28 | 0.972 | 0.956 |
| 3 | 10000 | 1000 | 7.4 | 2.1 | 1.42 | 1.54 | 0.958 | 0.954 |
| 4 | 100000 | 1000 | 27.8 | 2.5 | 1.67 | 1.76 | 0.952 | 0.951 |
| 5 | 1000000 | 1000 | 200.2 | 3.1 | 1.85 | 1.93 | 0.951 | 0.952 |

by using additional out-of-sample testing (with $N = 10^6$), that the average empirical probability of constraint satisfaction for the solutions generated by Y&dH is actually much higher at 0.984. The final solutions provided by ROBIST are less conservative and closer to the target, with an average empirical probability of 0.966.

4.1.2. Analysis of ROBIST.

In this subsection, we analyze ROBIST on a slightly altered version of our toy problem. We add an additional constraint ($1 + \sum_{j=1}^{k-1} x_j \leq x_k$) to (32)-(34), which allows us to analytically derive the true probability that (33) is satisfied:

$$p^*(\mathbf{x}) := \mathbb{P}(\tilde{\mathbf{z}}^\top \mathbf{x} \leq 1) = \frac{1}{2} + \frac{1}{2x_k}. \quad (35)$$

Furthermore, to expand the feasible region of the problem, we slightly alter Constraint (34), which becomes $\mathbf{x} \leq k$. Therefore, given knowledge of the true distribution of $\tilde{\mathbf{z}}$, one could solve the following optimization problem:

$$\theta^* := \max_{\mathbf{x}} \left\{ \mathbf{e}^\top \mathbf{x} : \frac{1}{2} + \frac{1}{2x_k} \geq 1 - \epsilon, \mathbf{x} \leq k, 1 + \sum_{j=1}^{k-1} x_j \leq x_k \right\}, \quad (36)$$

to obtain an optimal, sufficiently robust, solution. In the following sets of experiments we utilize (35) and (36) to assess the robustness and optimality of solutions obtained via Algorithm 1.

In the first set of experiments, we analyze the impact of the total amount of available data N . We do this for a problem setting with $k = 2$ and $\epsilon = 0.05$. Using $\alpha = 0.10$ and $i_{\max} = 1,000$. We apply ROBIST to the (altered) toy problem and for each iteration i , we store each obtained solution \mathbf{x}_i along with its proxy certificate $\tilde{\gamma}_i$ (obtained using the validation data). We repeat this procedure 100 times and evaluate the following three metrics:

1. optimality gap of the best found “sufficiently robust” solution:

$$\frac{\theta^* - \max_i \{g(\mathbf{x}_i) : \tilde{\gamma}_i \geq 1 - \epsilon, p^*(\mathbf{x}_i) \geq 1 - \epsilon\}}{\theta^*};$$

2. mean absolute error (MAE) of the proxy feasibility certificates:

$$\frac{1}{i_{\max}} \sum_{i=1}^{i_{\max}} |p^*(\mathbf{x}_i) - \tilde{\gamma}_i|;$$

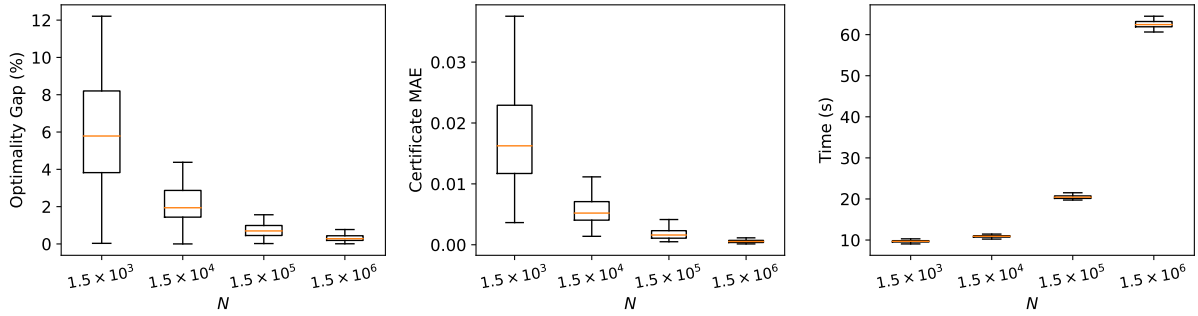


Figure 4: Box-and-whisker plots (with “outliers” omitted) displaying three metrics (optimality gap, certificate MAE and computation time) for ROBIST when applied to a slightly altered version of Problem (32)-(34) for $k = 2$. We plot the results as a function of the amount of available data (N).

3. total elapsed computation time of the algorithm.

The results are presented in Figure 4. For optimality gap and certificate MAE we see a very similar trend as N increases. While the two metrics might seem unrelated at first glance, the sharpness of the probability guarantees plays an important role in reducing conservativeness and thus closing the optimality gap.

We find that the computation time increases as N increases. However, the increase is modest in proportion to the increase in N and can be further reduced via parallelization. We infer from this analysis (along with other numerical experiments) that ROBIST is able to produce higher quality solutions and tighter probability guarantees when provided with more data, without inordinate increases in computation time.

In the second set of experiments, we analyze the efficiency of Algorithm 1 as the problem size k increases. Here we evaluate the following metrics:

1. i_{robust} : number of iterations required to find a sufficiently robust solution:

$$i_{\text{robust}} = \min\{i : p^*(\mathbf{x}_i) \geq 1 - \epsilon\};$$

2. i_{opt} : number of iterations required to obtain an optimality gap of less than 1%:

$$i_{\text{opt}} = \min\left\{i : \frac{\theta^* - g(\mathbf{x}_i)}{\theta^*} < 0.01, \tilde{\gamma}_i \geq 1 - \epsilon, p^*(\mathbf{x}_i) \geq 1 - \epsilon\right\};$$

3. $|\mathcal{S}_i|^{\max}$: maximum size of the scenario sets used to solve SCP_i :

$$|\mathcal{S}_i|^{\max} = \max_i\{|\mathcal{S}_i|\}.$$

We utilize the same setup as before ($\epsilon = 0.05$, $\alpha = 0.10$ and $i_{\max} = 1,000$), but with a fixed number of data points ($N = 1.5 \times 10^6$). The results of these experiments are displayed in Figure 5.

The most striking finding is that i_{robust} and $|\mathcal{S}_i|^{\max}$ both increase at similar, modest rates as the problem size increases. For example, while k increases 1000-fold, the maximum size of $|\mathcal{S}_i|$ becomes only 4.3 times as large (on average). We observe that 40 iterations is, in most cases, sufficient for ROBIST to obtain near optimal solutions for this problem setting.

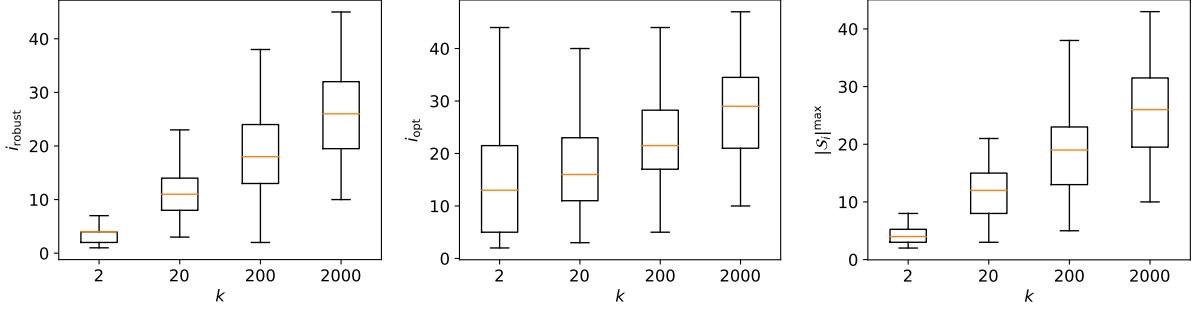


Figure 5: Box-and-whisker plots (with “outliers” omitted) displaying three metrics (i_{robust} , i_{opt} and $|\mathcal{S}_i|^{\text{max}}$) regarding the performance of ROBIST when applied to a slightly altered version of Problem (32)-(34) of varying size (k).

4.2. Linear Problem from Jiang and Xie (2022)

In this section, we compare the performance of ROBIST with methods from the sample average approximation literature. In this comparison, we consider the Linear CCP proposed by Jiang and Xie (2022), which is formulated as:

$$\begin{aligned} \max_{\mathbf{x} \in [0,1]^k} \quad & \mathbf{c}^\top \mathbf{x} \\ \text{s.t.} \quad & \mathbb{P}(\mathbf{a}^\top \mathbf{x} \leq 100) \geq 1 - \epsilon, \end{aligned} \tag{37}$$

where \mathbf{a} is an uncertain parameter uniformly distributed on the integers 1 to 50, and \mathbf{c} is a known vector, pre-generated randomly and uniformly on the integers 1 to 10.

4.2.1. Numerical Results.

We emulate the numerical experiments reported by Jiang and Xie (2022) and set $\epsilon = 0.1$. Furthermore, we set $\alpha = 0.01$, $i_{\text{max}} = 200$ and compare the performance of ROBIST with three alternative methods that can be used to solve the SAA variant of (37) for a given number of scenarios N . These three methods are: (i) a mixed-integer reformulation (abbreviated as MIO), (ii) ALSO-X and (iii) ALSO-X+. For the latter two methods we copy the implementation details proposed in Jiang and Xie (2022), see <https://github.com/jnan97/ALSO-X> for the code. To prevent excessive computation times for the MIO method, we set a time limit equal to the running time of ROBIST for the corresponding instance. The results are presented in Table 2.

We find that no single method dominates in terms of both objective value and probability of being feasible. ALSO-X+ is able to achieve the highest objective values. However, its solutions do not achieve the desired level of robustness ($1 - \epsilon = 0.9$). This is a consequence of the fact that SAA can overfit the data. Indeed, according to Theorem 10 of Luedtke and Ahmed (2008), to achieve an (out-of-sample) feasibility guarantee ≥ 0.9 for the problem instance with $k = 100$ would require $N > 10^6$. On the other hand, we find that ALSO-X is the most consistent method for providing solutions that achieve the desired level of robustness. However, in terms of objective value, its solutions are generally not as good as the solutions produced by ROBIST. The MIO approach, when given a time limit equal to ROBIST’s runtime, is not able to effectively solve instances with $N \geq 10,000$. For $N = 50,000$, simply finding a feasible solution using MIO takes two orders of magnitude longer than ROBIST. For ROBIST (with $i_{\text{max}} = 200$), we find that $N = 1,000$ is not sufficient to achieve the desired level of robustness, due to the high dimensionality $k \geq 100$. However, for $N \geq 5,000$, ROBIST is able to consistently provide

Table 2: Comparison between MIO, ALSO-X, ALSO-X+ and ROBIST for solving a random realization of Problem (37). The column “ $\mathbb{P}(\text{feas.})$ ” shows an empirical estimation of the probability that a solution is feasible (using 10^6 additional data points). For MIO, Gurobi is given a time limit equal to the running time of ROBIST for the instance (the actual runtime of MIO is longer, as the solver continues until it finds a feasible solution). For ROBIST, the feasibility certificates are also displayed in the final column “Cert.”

| k | N | MIO | | | ALSO-X | | | ALSO-X+ | | | ROBIST | | | |
|-----|-------|----------|-------|----------------------------|----------|-------|----------------------------|----------|-------|----------------------------|----------|-------|----------------------------|-------|
| | | Time (s) | Obj. | $\mathbb{P}(\text{feas.})$ | Time (s) | Obj. | $\mathbb{P}(\text{feas.})$ | Time (s) | Obj. | $\mathbb{P}(\text{feas.})$ | Time (s) | Obj. | $\mathbb{P}(\text{feas.})$ | Cert. |
| 100 | 1000 | 16.0 | 33.28 | 0.892 | 14.4 | 32.16 | 0.941 | 69.6 | 34.03 | 0.864 | 14.6 | 32.72 | 0.879 | 0.830 |
| | 5000 | 40.9 | 31.32 | 0.964 | 75.3 | 32.06 | 0.957 | 400.5 | 33.71 | 0.892 | 16.6 | 32.24 | 0.915 | 0.894 |
| | 10000 | 106.6 | 11.05 | 1.000 | 153.0 | 31.90 | 0.960 | 780.7 | 33.61 | 0.897 | 15.7 | 32.26 | 0.913 | 0.892 |
| | 50000 | 2006.5 | 10.29 | 1.000 | 952.4 | 31.84 | 0.958 | 4352.0 | 33.50 | 0.897 | 17.4 | 32.17 | 0.905 | 0.898 |
| 200 | 1000 | 22.4 | 34.17 | 0.908 | 36.3 | 33.77 | 0.927 | 189.6 | 35.63 | 0.829 | 20.2 | 34.26 | 0.860 | 0.820 |
| | 5000 | 71.2 | 32.80 | 0.967 | 162.8 | 33.39 | 0.954 | 858.2 | 34.97 | 0.887 | 25.8 | 33.65 | 0.914 | 0.898 |
| | 10000 | 190.6 | 10.33 | 1.000 | 298.8 | 33.43 | 0.955 | 1616.6 | 34.96 | 0.890 | 22.9 | 33.12 | 0.906 | 0.891 |
| | 50000 | 3590.8 | 10.29 | 1.000 | 1802.4 | 33.37 | 0.959 | 8264.4 | 34.79 | 0.898 | 27.7 | 33.37 | 0.908 | 0.902 |
| 300 | 1000 | 29.1 | 34.28 | 0.932 | 43.2 | 34.58 | 0.913 | 241.1 | 36.42 | 0.815 | 26.4 | 35.59 | 0.819 | 0.772 |
| | 5000 | 82.6 | 33.77 | 0.969 | 223.2 | 34.37 | 0.955 | 1322.4 | 35.67 | 0.888 | 30.3 | 34.82 | 0.900 | 0.873 |
| | 10000 | 219.8 | 10.38 | 1.000 | 496.6 | 34.43 | 0.953 | 2756.3 | 35.76 | 0.883 | 29.8 | 34.59 | 0.905 | 0.887 |
| | 50000 | 4675.5 | 10.29 | 1.000 | 2732.2 | 34.35 | 0.959 | 14982.2 | 35.57 | 0.898 | 37.5 | 34.37 | 0.904 | 0.900 |

solutions of adequate robustness within relatively stable computation times. More importantly, from the results presented in Table 2 we observe that the computation times of these three alternative methods do not scale as well as ROBIST as the amount of data N increases. Therefore, we infer that for larger scale problems, where one expects to require large amounts of data in order to adequately approximate the chance-constraint, ROBIST is more computationally tractable than these three alternative methods.

4.3. Weighted Distribution Problem

In the next set of experiments, we consider the weighted distribution problem of Carè et al. (2014). Suppose a company is able to produce and sell n different products with the usage of m different machines. The goal is to determine an optimal production plan, which specifies the amount of time x_{jk} that each machine $j = 1, \dots, m$ will be used for producing product $k = 1, \dots, n$. An optimal plan is one that maximizes the total profit of the company subject to availability constraints.

Each machine j may only be used for a limited amount of time a_j and incurs operating costs c_{jk} per unit of product k that is produced. Each unit of product k can be sold at a price of u_k and the leftover units incur holding costs h_k . For this problem, there are the following uncertain parameters: the demand \tilde{d}_k for each product k and the quantity \tilde{p}_{jk} of product k that is produced per allocated unit of time for machine j . The optimization problem is formulated as follows:

$$\max_{\mathbf{x}} \sum_{k=1}^n u_k \min \left\{ \sum_{j=1}^m p_{jk} x_{jk}, d_k \right\} - \sum_{j=1}^m \sum_{k=1}^n c_{jk} x_{jk} - \sum_{k=1}^n h_k \max \left\{ \sum_{j=1}^m p_{jk} x_{jk} - d_k, 0 \right\} \quad (38)$$

$$\text{s.t.} \quad \sum_{k=1}^n x_{jk} \leq a_j, \quad j = 1, \dots, m, \quad (39)$$

$$x_{jk} \geq 0, \quad j = 1, \dots, m, \quad k = 1, \dots, n. \quad (40)$$

We note that this is a difficult problem to deal with using conventional robust optimization techniques, since (38) is not convex in the uncertain parameter vectors $\tilde{\mathbf{d}}$ and $\tilde{\mathbf{p}}$.

For this problem, one is interested in obtaining a feasible and profitable production plan \mathbf{x} . However, due to the uncertainty in the demand of the products and the productivity of the machines, the exact profit can not be computed ahead of time. As in the portfolio management problem discussed in Appendix C.2 and the two-stage lot-sizing problem discussed in Appendix C.3, the uncertain parameters occur only in the objective. Hence, Algorithm 1 is slightly altered (see Section 3.1 for the details).

For this problem, we are interested in robust solutions for which one can state with confidence of at least $1 - \alpha$ that, if implemented, the realized profit will be larger than some threshold value with probability of at least $1 - \epsilon$. In other words, our objective is to maximize this threshold value (i.e., the Value-at-Risk), which represents a probabilistic lower bound on the realized profit.

4.3.1. Numerical Results.

We emulate the numerical experiments reported by Carè et al. (2014), where $\epsilon = 0.01$ and $\alpha = 10^{-9}$. The demand \tilde{d}_j is drawn from a Dirichlet distribution and the efficiency parameters \tilde{p}_{jk} are assumed to be uniformly distributed around some nominal values \bar{p}_{jk} with a $\pm 5\%$ maximum deviation. We refer to Carè et al. (2014) for the exact nominal values associated with the original problem with $m = 5$ machines and $n = 10$ products. For the larger problem instances (where $m > 5$ and $n > 10$), the nominal values are slightly perturbed. We let these nominal values be uniformly distributed within $\pm 10\%$ of the original problem.

We compare the performance of ROBIST with four existing scenario optimization methods from the literature. These are: C&C (Calafiore and Campi, 2005), FAST (Carè et al., 2014), RSD (Calafiore, 2017) and ISO (Garatti et al., 2022).

We implement the methods with the following settings. For C&C we utilize Theorem 1 from Campi and Garatti (2008) to determine the number of randomly sampled scenarios $N^{C\&C}$ with which (SCP) is solved. For FAST we follow the suggested rule of thumb to select the number of scenarios N_1^{FAST} with which (SCP) is solved (e.g., $N_1^{FAST} = 20mn$). For RSD, we set $\epsilon' = 0.7\epsilon$, determine N^{RSD} by requiring that the asymptotic upper bound on the expected number of iterations is less than or equal to 10 and then use Equation (18) in Calafiore (2017) to determine N_o^{RSD} . For ISO we use Algorithm 2 of Garatti et al. (2022) to determine the set sizes $N_0^{ISO}, N_1^{ISO}, \dots, N_{mn}^{ISO}$. Finally, for ROBIST we allow access to $N = 3,000$ data points and use a maximum of 200 iterations as stopping criteria. The results are reported for a single instance in Tables 3 and 4.

In Table 3 we find that, for the largest problem instance ($m = 15, n = 30$), the 10-hour time limit was reached for C&C, RSD and ISO. This is due to having to solve (SCP) with large sets of scenarios \mathcal{S} . By design, ROBIST utilizes significantly fewer scenarios when solving (SCP), which is clearly observed in the results corresponding to $|\mathcal{S}|^{max}$. This key difference enables ROBIST to remain computationally tractable when applied to the larger problem instances.

Next, in Table 4 we inspect the quality of the resulting solutions. We find that the four methods perform similarly in terms of the out-of-sample 1%-VaR. However, ROBIST, while having access to relatively few data points, is able to outperform the existing methods in terms of the objective value. Note that, in many real-world situations there may be a limited amount of data available and one may not have access to additional out-of-sample data. In such a situation one can only consult the objective value in order to determine the quality of a solution.

Table 3: Comparison between Calafiore and Campi (2005); Carè et al. (2014); Calafiore (2017); Garatti et al. (2022) and ROBIST in terms of the amount of data used (N), the maximum number of scenarios with which (SCP) is solved ($|\mathcal{S}|^{max}$) and the required computation time when applied to Problem (38)-(40) with varying number of machines m and products n . A dash (-) signifies that the time limit was reached before the relevant metric could be computed.

| m | n | N | | | | | $ \mathcal{S} ^{max}$ | | | | | Computation time (s) | | | | |
|-----|-----|-------|-------|-------|------|--------|-----------------------|------|-------|------|--------|----------------------|------|---------|---------|--------|
| | | C&C | FAST | RSD | ISO | ROBIST | C&C | FAST | RSD | ISO | ROBIST | C&C | FAST | RSD | ISO | ROBIST |
| 5 | 10 | 10580 | 3062 | 28662 | 5678 | 3000 | 10580 | 1000 | 6017 | 5678 | 65 | 699 | 10 | 239 | 17367 | 35 |
| 10 | 20 | 34918 | 6073 | 41733 | - | 3000 | 34918 | 4000 | 26160 | - | 86 | 27541 | 311 | 12438 | > 36000 | 76 |
| 15 | 30 | 74468 | 11073 | - | - | 3000 | 74468 | 9000 | 60586 | - | 96 | > 36000 | 3130 | > 36000 | > 36000 | 130 |

Table 4: Comparison between Calafiore and Campi (2005); Carè et al. (2014); Calafiore (2017); Garatti et al. (2022) and ROBIST in terms of the objective value and out-of-sample performance of the resulting solutions when applied to Problem (38)-(40) with a varying number of machines m and products n . A dash (-) signifies that the time limit was reached before a solution was found. The out-of-sample results are computed using 10^6 additional randomly generated scenarios.

| m | n | Objective value | | | | | Out-of-sample 1%-VaR | | | | |
|-----|-----|-----------------|--------|-------|-------|--------|----------------------|--------|-------|-------|--------|
| | | C&C | FAST | RSD | ISO | ROBIST | C&C | FAST | RSD | ISO | ROBIST |
| 5 | 10 | 457.8 | 441.4 | 464.7 | 458.7 | 455.3 | 473.5 | 477.3 | 476.3 | 471.7 | 475.0 |
| 10 | 20 | 973.2 | 945.5 | 974.6 | - | 962.1 | 1000.0 | 998.5 | 997.5 | - | 996.8 |
| 15 | 30 | - | 1469.7 | - | - | 1491.7 | - | 1514.7 | - | - | 1519.1 |

5. Conclusion

In this paper we propose ROBIST, a versatile, simple, data-driven and effective algorithm for dealing with optimization problems with uncertain parameters. A key element in ROBIST is the evaluation procedure, where probability guarantees are derived a posteriori using statistical testing. This procedure provides sharp probability guarantees that can be computed very efficiently, which allows the algorithm to identify and avoid overly conservative solutions. ROBIST can be applied to a wide variety of problem types and offers a number of practical advantages over existing methods. Furthermore, numerical experiments across a variety of applications show that ROBIST outperforms many alternative methods in terms of computational tractability as well as solution quality.

It is important to note that the probabilistic guarantees provided by the evaluation procedure are based on asymptotics (as $N \rightarrow \infty$) and are therefore only approximately valid. As such, ROBIST performs best when there is a large amount of data available.

An interesting direction for future research is the extension of ROBIST to treat multiple uncertain constraints (or objectives) separately. While the evaluation and generation procedures of ROBIST remain applicable, certain modifications, such as utilizing constraint-specific scenario sets, could improve the performance of the algorithm.

Appendix A. ϕ -divergence and confidence set

In order to formally evaluate the robustness of solutions, we use statistical testing that is based on ϕ -divergences. Given two vectors $\mathbf{p}, \mathbf{q} \in \mathbb{R}^{d_{\mathbf{p}}}$, a ϕ -divergence is defined as

$$I_{\phi}(\mathbf{q}, \mathbf{p}) = \sum_{i=1}^{d_{\mathbf{p}}} p_i \phi\left(\frac{q_i}{p_i}\right),$$

where $\phi : [0, \infty) \rightarrow \mathbb{R}$ is a convex function satisfying $\phi(1) = 0$, $\phi(a/0) := a \lim_{t \rightarrow \infty} \phi(t)/t$ for $a > 0$ and $\phi(0/0) = 0$. Using the modified χ^2 -distance, as we do throughout this paper, corresponds to choosing $\phi(t) = (t - 1)^2$. An extensive study of the statistical properties of ϕ -divergences, as well as an overview of common choices of $\phi(\cdot)$ functions, are given in Pardo (2006) and Ben-Tal et al. (2013).

In this paper we utilize the following property. Suppose \mathbf{p}^* is a probability vector and N data points are used to estimate \mathbf{p}^* with the empirical estimator $\hat{\mathbf{p}}$. Then, Pardo (2006) has shown that the following statistic:

$$\frac{2N}{\phi''(1)} I_{\phi}(\mathbf{p}^*, \hat{\mathbf{p}}),$$

converges (as $N \rightarrow \infty$) to a chi-squared distribution with $d_{\mathbf{p}} - 1$ degrees of freedom. Here, $\phi''(1)$ denotes the second derivative of ϕ evaluated at 1. Hence, one can construct the following $(1 - \alpha)$ -confidence set for the true probability vector \mathbf{p}^* , as a ϕ -divergence ball around the empirical estimate $\hat{\mathbf{p}}$:

$$\left\{ \mathbf{q} \in \mathbb{R}^{d_{\mathbf{p}}} : \mathbf{q} \geq 0, \mathbf{q}^T \mathbf{1} = 1, I_{\phi}(\mathbf{q}, \hat{\mathbf{p}}) \leq \frac{\phi''(1)}{2N} \chi_{d_{\mathbf{p}}-1, 1-\alpha}^2 \right\},$$

where $\chi_{d_{\mathbf{p}}-1, 1-\alpha}^2$ is the $(1 - \alpha)$ -quantile of the chi-squared distribution with degree $d_{\mathbf{p}} - 1$.

Appendix B. Proofs

Appendix B.1. Proof of Lemma 1

By definition, we have that

$$\gamma_i = \min_{q \geq 0} \left\{ q : p \left(\frac{q}{p} - 1 \right)^2 + (1 - p) \left(\frac{1 - q}{1 - p} - 1 \right)^2 \leq r \right\},$$

where $r = \frac{1}{N} \chi_{1, 1-\alpha}^2$ and p is an empirical estimate based on N independent observations. Since the objective function is linear and the constraints are convex, we can determine the optimal solution by solving the following quadratic equation:

$$p \left(\frac{q}{p} - 1 \right)^2 + (1 - p) \left(\frac{1 - q}{1 - p} - 1 \right)^2 = r.$$

Solving this for q yields the smallest solution $q = p - \sqrt{p(1-p)r}$. Since the constraint $q \geq 0$ must also hold, we have that $\gamma_i = \max\{p - \sqrt{p(1-p)r}, 0\}$. To show that γ_i is also increasing in p , we first note that the function $p \mapsto p - \sqrt{p(1-p)r}$ is convex in p , and thus is increasing after its minimum. Furthermore, we have

$$p - \sqrt{p(1-p)r} \geq 0 \Leftrightarrow p \geq \frac{r}{1+r}.$$

Hence, its minimum, which is smaller than zero, can only be attained for $p < \frac{r}{1+r}$. Therefore, $\gamma_i > 0$ only if $p \geq \frac{r}{1+r}$ and thus γ_i is increasing in p .

Appendix B.2. Proof of Lemma 2

Let $\{\check{\mathcal{S}}_0, \dots, \check{\mathcal{S}}_M\}$ be all the possible subsets of the training data set $\mathcal{D}_{N_1}^{\text{train}} = \{\hat{\mathbf{z}}^1, \dots, \hat{\mathbf{z}}^{N_1}\}$ with $\check{\mathcal{S}}_0 = \emptyset$ and $M = 2^{N_1} - 1$. For each subset $\check{\mathcal{S}}_j \in \{\check{\mathcal{S}}_0, \dots, \check{\mathcal{S}}_M\}$, we denote $\check{\mathbf{x}}_j^*$ as the unique optimal solution of the corresponding Problem (SCP_{*i*}), where $\mathcal{S}_i = \check{\mathcal{S}}_j$. Furthermore, we denote $\hat{\gamma}(\check{\mathbf{x}}_j^*)$ as the certificate of $\check{\mathbf{x}}_j^*$, derived using $\mathcal{D}_{N_1}^{\text{train}}$. Finally, we define the corresponding ‘‘infeasibility’’ set $\text{IFeas}(\check{\mathbf{x}}_j^*)$ as:

$$\text{IFeas}(\check{\mathbf{x}}_j^*) := \{\hat{\mathbf{z}}^j \in \mathcal{D}_{N_1}^{\text{train}} : f(\check{\mathbf{x}}_j^*, \hat{\mathbf{z}}^j) > 0\}.$$

Letting \mathcal{S}_i denote the subset used during the *i*-th iteration of Algorithm 1 and following the addition and removal procedure described in Section 2, we have the following transition probabilities between the possible subsets $\{\check{\mathcal{S}}_1, \dots, \check{\mathcal{S}}_M\}$:

$$\mathbb{P}(\mathcal{S}_{i+1} = \check{\mathcal{S}}_j \mid \mathcal{S}_i = \check{\mathcal{S}}_k) = \begin{cases} (1 - v) \cdot \frac{1}{|\text{IFeas}(\check{\mathbf{x}}_k^*)|} & \text{if } \hat{\gamma}(\check{\mathbf{x}}_k^*) \leq 1 - \epsilon \text{ and } \check{\mathcal{S}}_j = \check{\mathcal{S}}_k \cup \{\hat{\mathbf{z}}\}, \hat{\mathbf{z}} \notin \check{\mathcal{S}}_k, \\ v \cdot \frac{1}{|\text{IFeas}(\check{\mathbf{x}}_k^*)|} & \text{if } \hat{\gamma}(\check{\mathbf{x}}_k^*) > 1 - \epsilon \text{ and } \check{\mathcal{S}}_j = \check{\mathcal{S}}_k \cup \{\hat{\mathbf{z}}\}, \hat{\mathbf{z}} \notin \check{\mathcal{S}}_k, \\ (1 - v) \cdot \frac{1}{|\check{\mathcal{S}}_k|} & \text{if } \hat{\gamma}(\check{\mathbf{x}}_k^*) \geq 1 - \epsilon \text{ and } \check{\mathcal{S}}_j = \check{\mathcal{S}}_k \setminus \{\hat{\mathbf{z}}\}, \hat{\mathbf{z}} \in \check{\mathcal{S}}_k, \\ v \cdot \frac{1}{|\check{\mathcal{S}}_k|} & \text{if } \hat{\gamma}(\check{\mathbf{x}}_k^*) < 1 - \epsilon \text{ and } \check{\mathcal{S}}_j = \check{\mathcal{S}}_k \setminus \{\hat{\mathbf{z}}\}, \hat{\mathbf{z}} \in \check{\mathcal{S}}_k, \\ 0 & \text{otherwise.} \end{cases}$$

Since the transition probability depends only on the previous subset, we have that \mathcal{S}_i constitutes a time-homogeneous Markov chain with finitely many states. This finiteness implies that there exists at least one particular subset for which the corresponding solution $\check{\mathbf{x}}^*$ is optimal with respect to the test data.

We will now show that for all possible subsets/states, there is a path with positive probability to one of the subsets with the optimal solution. Indeed, for any subset $\check{\mathcal{S}}_j$, there is always a probability of removal and hence a path to the empty set $\check{\mathcal{S}}_0$, which we denote as $\check{\mathcal{S}}_j \rightarrow \check{\mathcal{S}}_0$.

Let \mathcal{S}^{opt} be the collection of all optimal subsets and let $\check{\mathcal{S}}_{k^*} \in \mathcal{S}^{\text{opt}}$ be a particular optimal subset. We claim that there is a path from $\check{\mathcal{S}}_0$ to the class \mathcal{S}^{opt} :

$$\check{\mathcal{S}}_0 \rightarrow \dots \rightarrow \mathcal{S}^{\text{opt}}.$$

Indeed, let $\check{\mathbf{x}}_0^*$ be the solution of the empty set $\check{\mathcal{S}}_0$. Since $\max_{\mathbf{x} \in \mathcal{X}} \{g(\mathbf{x})\} \geq \max_{\mathbf{x} \in \mathcal{X}} \{g(\mathbf{x}) : f(\mathbf{x}, \hat{\mathbf{z}}^j) \leq 0, \forall \hat{\mathbf{z}}^j \in \check{\mathcal{S}}_{k^*}\}$, we have that if $\check{\mathbf{x}}_0^*$ is feasible for all scenarios in $\check{\mathcal{S}}_{k^*}$, then $\check{\mathbf{x}}_0^*$ must also be the unique optimal solution (uniqueness by assumption or by Remark 2) of solving (SCP_{*i*}) with $\mathcal{S}_i = \check{\mathcal{S}}_{k^*}$. In that case, we have by definition that $\check{\mathbf{x}}_0^*$ is an optimal solution with respect to the test data and thus implies $\check{\mathcal{S}}_0 \in \mathcal{S}^{\text{opt}}$. Therefore, without loss of generality, we may assume that $\check{\mathbf{x}}_0^*$ is infeasible for at least one scenario, say $\hat{\mathbf{z}}^q$ of $\check{\mathcal{S}}_{k^*}$. Since there is a positive probability of adding this scenario, there is a positive probability path $\check{\mathcal{S}}_0 \rightarrow \check{\mathcal{S}}_0 \cup \{\hat{\mathbf{z}}^q\}$ and we can now repeat the same argument above for the solution $\check{\mathbf{x}}^*$ of $\check{\mathcal{S}}_0 \cup \{\hat{\mathbf{z}}^q\}$: if $\check{\mathbf{x}}^*$ is feasible for all scenarios in $\check{\mathcal{S}}_{k^*}$, then $\check{\mathcal{S}}_0 \cup \{\hat{\mathbf{z}}^q\} \in \mathcal{S}^{\text{opt}}$. Otherwise, there is a positive probability of adding another scenario of $\check{\mathcal{S}}_{k^*}$. This argument continues until either the subset $\check{\mathcal{S}}_{k^*}$ is reached, or an optimal subset is reached earlier in the path. Thus, there is a positive probability path to all subsets in the Markov chain. Therefore, the Markov chain is time-homogeneous and irreducible, which implies that the hitting probability of any state is 1 (Norris, 1997, Theorem 5.8).

Appendix C. Extra numerical experiments

Appendix C.1. Altered Toy Problem

Our altered toy problem is formulated as follows:

$$\max_{\mathbf{x}} \mathbf{e}^\top \mathbf{x} \tag{C.1}$$

$$\text{s.t. } \tilde{\mathbf{z}}^\top \mathbf{x} \leq 1, \tag{C.2}$$

$$1 + \sum_{j=1}^{k-1} x_j \leq x_k, \tag{C.3}$$

$$\mathbf{x} \leq k, \tag{C.4}$$

where $\mathbf{x}, \mathbf{z} \in \mathbb{R}^k$, and $\mathbf{e}^\top = (1, 1, \dots, 1) \in \mathbb{R}^k$. We assume that the uncertain parameter vector \mathbf{z} is stochastic, where the random variables $\tilde{z}_1, \dots, \tilde{z}_k$ are assumed to be independently and uniformly distributed in $[-1, 1]$. We are interested in obtaining a robust solution to Problem (C.1)-(C.4), i.e., a solution that satisfies Constraint (C.2) with probability greater than or equal to $1 - \epsilon$.

In the following paragraphs we use Problem (C.1)-(C.4) as a benchmark for evaluating certain components of ROBIST, as presented in Algorithm 1. Recall that for this problem we are able to analytically derive the true probability that (C.2) is satisfied:

$$p^*(\mathbf{x}) := \mathbb{P}(\tilde{\mathbf{z}}^\top \mathbf{x} \leq 1) = \frac{1}{2} + \frac{1}{2x_k}. \tag{C.5}$$

This enables an analysis of the following three metrics:

1. i_{robust} : number of iterations required to find a sufficiently robust solution:

$$i_{\text{robust}} = \min\{i : p^*(\mathbf{x}_i) \geq 1 - \epsilon\};$$

2. $|\mathcal{S}_i|^{\max}$: maximum size of the scenario sets used to solve SCP_{*i*}:

$$|\mathcal{S}_i|^{\max} = \max_i \{|\mathcal{S}_i|\};$$

3. optimality gap of the best found sufficiently robust solution:

$$\frac{\theta^* - \max_i \{g(\mathbf{x}_i) : \check{\gamma}_i \geq 1 - \epsilon, p^*(\mathbf{x}_i) \geq 1 - \epsilon\}}{\theta^*}.$$

Appendix C.1.1. Effect of v .

First, we examine the input parameter v . Recall that this parameter controls the probability of reversing the addition or removal action suggested by evaluation on the training data. As stated earlier, this random component is added to ensure theoretical convergence of Algorithm 1 and for the other numerical experiments presented in this paper it is set to 0.01. Here we consider alternative choices for v , where we vary $v \in \{0.01, 0.25, 0.5\}$ and analyze the performance of ROBIST under such a parameter setting. Note that setting $v = 0.5$ is equivalent to a strategy where, at each iteration, a scenario is added or removed with equal probability.

Our setup is as follows: $\epsilon = 0.05, \alpha = 0.01, i_{\max} = 200$, where we vary $k \in \{20, 200, 2000\}$ and $N \in \{3000, 30000\}$. To control for the effects of randomness, we repeat the experiment 10 times and report the average. The results are presented in Table C.5.

Table C.5: Analysis of impact of parameter v on performance of ROBIST when applied to Problem (C.1)-(C.4). Settings: $\epsilon=0.05$, $\alpha=0.01$ and $i_{max}=200$. The results are averaged over 10 repetitions.

| k | N | v | i_{robust} | | | $ \mathcal{S}_i ^{\text{max}}$ | | | Opt. Gap (%) | | |
|------|-------|-----|---------------------|------|-------|--------------------------------|------|------|--------------|------|-------|
| | | | 0.01 | 0.25 | 0.50 | 0.01 | 0.25 | 0.50 | 0.01 | 0.25 | 0.50 |
| 20 | 3000 | | 9.4 | 6.3 | 11.7 | 14.1 | 13.9 | 10.3 | 6.01 | 6.52 | 6.97 |
| | 30000 | | 9.6 | 12.6 | 20.8 | 13.5 | 13.9 | 15.9 | 2.14 | 2.40 | 3.74 |
| 200 | 3000 | | 17.2 | 18.6 | 56.2 | 20.7 | 18.4 | 16.9 | 6.26 | 6.32 | 16.29 |
| | 30000 | | 18.5 | 22.9 | 50.8 | 21.4 | 18.8 | 17.4 | 1.77 | 2.02 | 3.56 |
| 2000 | 3000 | | 24.0 | 30.4 | 118.6 | 27.2 | 25.6 | 16.5 | 5.43 | 6.06 | 25.44 |
| | 30000 | | 22.9 | 32.8 | 117.0 | 27.9 | 26.1 | 20.0 | 2.11 | 2.15 | 12.86 |

We find that a higher degree of randomization, i.e., $v \in \{0.25, 0.50\}$ instead of $v = 0.01$, generally performs worse in terms of i_{robust} and the optimality gap. While the performance of ROBIST with $v = 0.25$ is comparable to our default (0.01), the performance is significantly affected when v is set to 0.50. Here we find a large difference in terms of the number of iterations required to find a sufficiently robust solution. Interestingly, a higher degree of randomization does lead to relatively smaller scenario sets, which could be beneficial in terms of computational effort. However, this benefit is likely offset by the increased number of iterations required.

Overall, we conclude from these numerical experiments that setting $v \in [0, 0.25]$ is unlikely to have a significant impact on the performance of Algorithm 1.

Appendix C.1.2. Addition strategy.

Given the decision to add or remove a scenario from our sampled set \mathcal{S} , one could think of various ways to select a scenario to add/remove. In this section we examine the addition strategy of ROBIST.

Our default strategy, the strategy that is used throughout this paper, is to pick a random scenario from amongst the scenarios for which our current solution violates the constraint, i.e., at iteration i , we pick randomly from the set: $\{\hat{\mathbf{z}}^j \in \mathcal{D}_{N_1}^{\text{train}} : f(\mathbf{x}_i, \hat{\mathbf{z}}^j) > 0\}$. We abbreviate this strategy as ‘‘RV’’ (short for random violation). An alternative strategy is to simply select a random scenario from amongst the scenarios that are not already in our set, i.e., at iteration i , we pick randomly from the set: $\{\hat{\mathbf{z}}^j \in \mathcal{D}_{N_1}^{\text{train}} \setminus \mathcal{S}_i\}$. We abbreviate this strategy as ‘‘RA’’ (short for random any). Another alternative strategy is to simply select a random scenario from amongst the scenarios for which our current solution is maximally violated, i.e., at iteration i , we pick a random scenario from the set: $\{\hat{\mathbf{z}}^j \in \mathcal{D}_{N_1}^{\text{train}} : \hat{\mathbf{z}}^j \in \arg \max_l f(\mathbf{x}_i, \hat{\mathbf{z}}^l), f(\mathbf{x}_i, \hat{\mathbf{z}}^j) > 0\}$. We abbreviate this strategy as ‘‘MV’’ (short for maximal violation).

We compare these two alternative addition strategies with our default strategy by applying them to Problem (C.1)-(C.4). Our setup is as follows: $\epsilon = 0.05$, $\alpha = 0.01$, $i_{\text{max}} = 200$, where we vary $k \in \{20, 200, 2000\}$ and $N \in \{3000, 30000\}$. To control for the effects of randomness, we repeat the experiment 10 times and report the average. The results are presented in Table C.6.

We find that MV does best in terms of i_{robust} and $|\mathcal{S}_i|^{\text{max}}$. Intuitively speaking, MV is quicker to identify critical scenarios and add these to the scenario set. However, this

Table C.6: Analysis of impact of the addition strategy on performance of ROBIST when applied to Problem (C.1)-(C.4). Settings: $\epsilon=0.05$, $\alpha=0.01$, $i_{max}=200$. The results are averaged over 10 repetitions.

| k | N | Strat. | i_{robust} | | | $ \mathcal{S}_i ^{\max}$ | | | Opt. Gap (%) | | |
|------|-------|--------|--------------|-------|-----|--------------------------|-------|-----|--------------|-------|-------|
| | | | RV | RA | MV | RV | RA | MV | RV | RA | MV |
| 20 | 3000 | | 9.4 | 49.3 | 2.1 | 14.1 | 101.5 | 3.4 | 6.01 | 36.15 | 10.40 |
| | 30000 | | 9.6 | 85.9 | 1.8 | 13.5 | 129.3 | 3.0 | 2.14 | 5.55 | 8.99 |
| 200 | 3000 | | 17.2 | 108.6 | 3.9 | 20.7 | 145.9 | 5.8 | 6.26 | 35.08 | 6.70 |
| | 30000 | | 18.5 | 102.7 | 2.2 | 21.4 | 128.9 | 3.7 | 1.77 | 5.35 | 3.51 |
| 2000 | 3000 | | 24.0 | 120.6 | 5.6 | 27.2 | 157.2 | 7.3 | 5.43 | 53.52 | 5.44 |
| | 30000 | | 22.9 | 138.8 | 3.6 | 27.9 | 158.6 | 5.2 | 2.11 | 32.34 | 2.40 |

strategy may lead to overly conservative solutions, which is reflected in the results for the optimality gap, for which our default strategy (RV) outperforms the two alternatives.

Overall, we conclude from these numerical experiments that RV performs significantly better than RA and that MV is likely to be an effective strategy for large scale instances with small ϵ .

Appendix C.2. Portfolio Management Problem

In this subsection we apply ROBIST to a portfolio management problem. As in Bertsimas et al. (2018), we consider an uncertain single period allocation problem:

$$\max_{\mathbf{x} \geq \mathbf{0}} \mathbf{z}^\top \mathbf{x} \tag{C.6}$$

$$\text{s.t. } \mathbf{e}^\top \mathbf{x} = 1. \tag{C.7}$$

For this problem one seeks a profit-maximizing allocation $\mathbf{x} \in \mathbb{R}^k$ across k different assets, for which the returns $\mathbf{z} \in \mathbb{R}^k$ are uncertain.

Note that in this problem the uncertainty only affects the objective. As such, when applying ROBIST, Algorithm 1 is slightly altered (see Section 3.1 for the details). Rewriting (C.6)-(C.7) using an epigraph reformulation, we obtain the following:

$$\max_{\mathbf{x}, \theta} \theta \tag{C.8}$$

$$\text{s.t. } \mathbf{z}^\top \mathbf{x} \geq \theta, \tag{C.9}$$

$$\mathbf{e}^\top \mathbf{x} = 1, \tag{C.10}$$

$$\mathbf{x} \geq \mathbf{0}. \tag{C.11}$$

We note that solving (C.8)-(C.11) while providing a probability guarantee for Constraint (C.9) is equivalent to maximizing the value at risk (VaR), or $(100 \times \epsilon)\%$ -quantile, of the portfolio's profit, as:

$$\mathbb{P}(\tilde{\mathbf{z}}^\top \mathbf{x} \geq \theta) \geq 1 - \epsilon \iff \text{VaR}_\epsilon^{\mathbb{P}}(\tilde{\mathbf{z}}^\top \mathbf{x}) \leq \theta. \tag{C.12}$$

Appendix C.2.1. Numerical Results.

We follow Bertsimas et al. (2018) by utilizing the model from Natarajan et al. (2008) to synthetically generate returns for k assets. This is done for a single time period in the following manner:

$$z_i = \begin{cases} \frac{\sqrt{(1-\gamma_i)\gamma_i}}{\gamma_i} & \text{with probability } \gamma_i \\ -\frac{\sqrt{(1-\gamma_i)\gamma_i}}{1-\gamma_i} & \text{with probability } 1 - \gamma_i \end{cases}, \quad \gamma_i = \frac{1}{2} \left(1 + \frac{i}{k+1} \right), \quad i = 1, \dots, k. \quad (\text{C.13})$$

In this model, all assets $i = 1, \dots, k$ have mean return 0%, standard deviation 1%, but have different skew and support. The higher indexed assets have a larger γ_i and are more negatively skewed and thus more likely to generate large losses and small upside gains. The returns for the assets are assumed to be independent.

We evaluate the performance of ROBIST with $i_{max} = 200$ by comparing it to the results reported in Table 3 in Bertsimas et al. (2018) (where $k = 10$ and $\alpha = \epsilon = 0.1$). In Table C.7 we report the average out-of-sample 10%-VaR over 100 repetitions.

Table C.7: Average 10%-VaR on out-of-sample realized returns, computed using 10^6 additional randomly generated scenarios. ROBIST is compared with the methods presented in Table 3 of Bertsimas et al. (2018).

| N | M | LCX | CS | CM | ROBIST |
|------|--------|--------|--------|--------|--------|
| 500 | -1.095 | -0.411 | -0.397 | -0.539 | 0.194 |
| 2000 | -1.095 | -0.411 | -0.396 | -0.451 | 0.316 |

We find that ROBIST significantly outperforms the other solution methods of Shawe-Taylor and Cristianini (2003); Calafiore (2013) and Bertsimas et al. (2018) in regard to average out-of-sample performance. The large difference in performance is explained by the difference in portfolio holdings, which is displayed in Figure C.6. Here we find that the solutions found by ROBIST put all wealth in either asset 9 or 10.

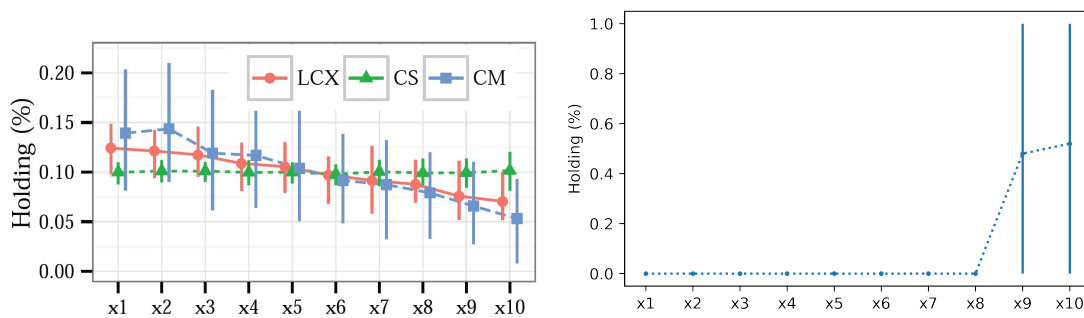


Figure C.6: Graphical display of the average, along with 10% and 90% quantiles, of the portfolio holdings by method across the 100 repetitions of the experiment with $N = 500$. On the left we show the portfolio holdings reported in Figure 4 of Bertsimas et al. (2018), on the right we display the same information for the portfolio holdings found by applying ROBIST.

Upon further inspection of the data generation procedure of Natarajan et al. (2008) for $k = 10$, one finds that $\gamma_9 = 0.909$ and $\gamma_{10} = 0.955$. This implies that:

$$\tilde{r}_9 = \begin{cases} 0.32 & \text{with probability } 0.909 \\ -3.16 & \text{with probability } 1 - 0.909 \end{cases} \quad \tilde{r}_{10} = \begin{cases} 0.22 & \text{with probability } 0.955 \\ -4.58 & \text{with probability } 1 - 0.955. \end{cases}$$

Thus, when $x_9 = 1$ the 10%-VaR is $= 0.32$, and when $x_{10} = 1$, the 10%-VaR is 0.22 .

While the allocations obtained via ROBIST are somewhat trivial and arguably risky, they do outperform the allocations found by the other methods in terms of the objective under consideration (10%-VaR). Our solutions exploit a flaw in optimizing for VaR when using the generation procedure of Natarajan et al. (2008) with $k = 10$ assets and $\epsilon = 10\%$. This flaw was not discovered by the other methods, which is due to their more conservative approach.

This flaw highlights a well-known danger in optimizing for VaR, namely that it does not account for the magnitude of losses that occur with probability less than ϵ . Alternatively one may consider optimizing the conditional VaR instead; see Basak and Shapiro (2001) and Laeven and Stajde (2014). We note that this is also possible with ROBIST (see Section 3.3).

Appendix C.3. Two-Stage Lot-Sizing Problem

In the final set of experiments, we evaluate the performance of ROBIST on a two-stage adaptive lot-sizing problem. A similar variant of this problem is studied in Bertsimas and de Ruiter (2016).

Consider a network of m nodes, where each node $i \in \{1, \dots, m\}$ has uncertain demand \tilde{d}_i . The demand at each node in the network must be satisfied and this can be done through the initial allocation of stock x_i , or by moving y_{ji} units of stock from node j to node i . The initial allocation of stock at node i costs c_i per unit, while the unit transportation costs are uncertain and denoted by \tilde{t}_{ij} . Each node has a maximum allocation capacity of k_i units.

We model this as a two-stage adaptive problem where the initial allocation decisions \mathbf{x} must be made before the uncertainty is realized. In the second stage, the transportation decisions y_{ij} can adapt to the realized demand and transportation costs. For notational ease, we denote uncertain parameters \tilde{d}_i and \tilde{t}_{ij} as a single vector $\tilde{\mathbf{z}} \in \mathbb{R}^{m+m^2}$ and formulate the problem as:

$$\min_{\mathbf{x}, \theta} \theta \tag{C.14}$$

$$\text{s.t. } \sum_{i=1}^m c_i x_i + V(\mathbf{z}, \mathbf{x}) \leq \theta, \tag{C.15}$$

$$0 \leq x_i \leq k_i, \quad i = 1, \dots, m, \tag{C.16}$$

where:

$$V(\mathbf{z}, \mathbf{x}) := \min_{\mathbf{y} \geq \mathbf{0}} \sum_{i=1}^m \sum_{j=1}^m \tilde{t}_{ij} y_{ij} \tag{C.17}$$

$$\text{s.t. } x_i + \sum_{j=1}^n y_{ji} - \sum_{j=1}^n y_{ij} \geq d_i, \quad i = 1, \dots, m. \tag{C.18}$$

If Constraint (C.18) is not satisfied, $V(\mathbf{z}, \mathbf{x}) = \infty$. For this problem, we are interested in solutions for which we can provide statistical guarantees of the form (5), with $\gamma \geq 1 - \epsilon$, for Constraint (C.15).

Finally, note that most techniques from RO, such as the method presented in Bertsimas and de Ruiter (2016), are unable to deal with adaptive problems with “random recourse” (i.e., when the adaptive decisions are multiplied with uncertain parameters). As \tilde{t}_{ij} is uncertain, the majority of existing RO methods can not be applied to this problem.

Appendix C.3.1. Numerical Results.

We replicate the parameter settings utilized by Bertsimas and de Ruiter (2016). However, instead of using an uncertainty set, we sample \tilde{d}_i and \tilde{t}_{ij} in the following manner:

1. Realizations of \tilde{d}_i are uniformly sampled from the budgeted uncertainty set described in Bertsimas and de Ruiter (2016) via the hit-and-run sampling method of Smith (1984).
2. Let the Euclidean distance from i to j be v_{ij} . We generate realizations of \tilde{t}_{ij} by uniformly sampling in the range $[0.9v_{ij}, 1.1v_{ij}]$.

In this set of experiments, we compare ROBIST with the solution approach of Vayanos et al. (2012) (hereafter abbreviated as VKR). The idea behind this approach is to approximate the adaptive decisions using “decision rules” (finite linear combinations of the uncertain parameters). This allows one to reduce a multistage adaptive problem to a single stage static problem. Then, one can apply the theory from Campi and Garatti (2008) to determine the necessary number of randomly sampled constraints in order to obtain solutions that satisfy the desired probability guarantee.

In our numerical experiments, we implement VKR using polynomial decision rules of degree p , where $p \in \{1, 2\}$. For such decision rules the adaptive decisions \mathbf{y} are substituted by linear combinations of a basis vector $\mathbf{b}(\tilde{\mathbf{z}}) \in \mathbb{R}^{s_p}$, where $s_p = \binom{m+m^2+p}{p}$ (we refer to Vayanos et al. (2012) for further details). Thus, $\mathbf{y} = A\mathbf{b}(\tilde{\mathbf{z}})$, where $A \in \mathbb{R}^{(m \times m) \times s_p}$ contains the coefficients of the linear combinations, which are treated as decision variables. This reduces Problem (C.14)-(C.18) to the following single-stage form:

$$\min_{\mathbf{x}, A, \theta} \theta \tag{C.19}$$

$$\text{s.t. } \sum_{i=1}^m c_i x_i + \sum_{i=1}^m \sum_{j=1}^m t_{ij} y_{ij} \leq \theta, \tag{C.20}$$

$$x_i + \sum_{j=1}^n y_{ji} - \sum_{j=1}^n y_{ij} \geq d_i, \quad i = 1, \dots, m, \tag{C.21}$$

$$\mathbf{y} = A\mathbf{b}(\tilde{\mathbf{z}}), \tag{C.22}$$

$$\mathbf{y} \geq \mathbf{0}, \tag{C.23}$$

$$0 \leq x_i \leq k_i, \quad i = 1, \dots, m. \tag{C.24}$$

Problem (C.19)-(C.24) can then be solved with respect to some set of randomly sampled scenarios (let this set be denoted as \mathcal{S}_{VKR}), where constraints (C.20)-(C.23) are duplicated for each scenario $\mathbf{z} \in \mathcal{S}_{VKR}$. The number of randomly sampled scenarios $|\mathcal{S}_{VKR}|$ is determined using Theorem 1 of Campi and Garatti (2008), which depends on ϵ , α and the number of decision variables $(1 + m + m^2 s_p)$.

The ROBIST algorithm is slightly modified when applied to adaptive optimization problems (see Section 3.2 for the details). An important aspect to note is that, at each iteration i , a new solution is generated by solving a problem in the form of (21)-(24), which involves $1 + m + m^2 |\mathcal{S}_i|$ decision variables instead of the original $1 + m + m^2$ variables used in defining Problem (C.14)-(C.18). In these experiments we set $N = 809$ (the minimum number of scenarios utilized by VKR) and $i_{max} = 50$.

Setting $\epsilon = \alpha = 0.05$, we compare the two approaches to Problem (C.14)-(C.18) as the number of nodes (m) increases. The numerical results (average over 10 replications) are presented in Tables C.8 and C.9.

In Table C.8 we find that VKR is faster than ROBIST for the small problem instances ($m \leq 3$). However, the required amount of randomly sampled scenarios ($|\mathcal{S}_{VKR}|$) and the resulting computation time rapidly increase as m increases. In comparison, we again find that ROBIST is effectively able to sample fewer scenarios (see $\max_i |\mathcal{S}_i|$), retaining computational tractability as the problem size increases.

Table C.8: Comparison between two implementations of Vayanos et al. (2012) (where $p = 1$ or $p = 2$) and ROBIST in terms of the amount of data used ($|\mathcal{S}_{VKR}|$ and N), the maximum number of scenarios with which (SCP) is solved ($|\mathcal{S}_{VKR}|$ and $\max_i |\mathcal{S}_i|$) and the required computation time when applied to Problem (C.14)-(C.18) with a varying number of nodes m .

| m | $ \mathcal{S}_{VKR} $ | | ROBIST | | Computation time (s) | | |
|-----|-----------------------|---------|--------|--------------------------|----------------------|--------------|--------|
| | $p = 1$ | $p = 2$ | N | $\max_i \mathcal{S}_i $ | VKR $_{p=1}$ | VKR $_{p=2}$ | ROBIST |
| 2 | 809 | 2655 | 809 | 9.0 | 5 | 60 | 196 |
| 3 | 2783 | 26103 | 809 | 10.7 | 56 | > 3600 | 204 |
| 4 | 10853 | 117162 | 809 | 15.1 | 2214 | > 3600 | 214 |
| 5 | 24774 | 392584 | 809 | 14.3 | > 3600 | > 3600 | 237 |

In Table C.9, we inspect the quality of the resulting solutions for the cases that a solution was obtained within the one hour time limit. In all tests the initial allocation was sufficient to satisfy the total realized demand, thus the average 5%-VaR of the out-of-sample realized costs provides a fair metric of comparison between the methods. Across all the conducted experiments we find that the solutions obtained via VKR are outperformed by the solutions obtained via ROBIST.

Table C.9: Comparison between two implementations of Vayanos et al. (2012) (where $p = 1$ or $p = 2$) and ROBIST in terms of the objective value and out-of-sample performance when applied to Problem (C.14)-(C.18) with a varying number of nodes m . A dash (-) signifies that the time limit was reached before a solution was found. The out-of-sample results are computed using 10^4 additional randomly generated scenarios.

| m | Objective value | | | Out-of-sample 5%-VaR | | |
|-----|-----------------|--------------|--------|----------------------|--------------|--------|
| | VKR $_{p=1}$ | VKR $_{p=2}$ | ROBIST | VKR $_{p=1}$ | VKR $_{p=2}$ | ROBIST |
| 2 | 795 | 792 | 736 | 788 | 778 | 731 |
| 3 | 1192 | - | 1041 | 1178 | - | 1021 |
| 4 | 1588 | - | 1229 | 1555 | - | 1206 |
| 5 | - | - | 1386 | - | - | 1348 |

Acknowledgments

The authors are very grateful to İhsan Yanıkoğlu, Dimitris Bertsimas, Vishal Gupta, Daniel Kuhn and Simone Garatti for their cooperation in setting up the numerical experiments presented in this paper. The authors also extend gratitude to the anonymous reviewers whose thoughtful comments led to significant improvements to the manuscript. This research was funded in part by the Netherlands Organization for Scientific Research under grants NWO VICI 2020-2027 (Laeven and Jin) and ESI.2019.003 (den Hertog and Starreveld).

References

- Ahmed, S., Luedtke, J., Song, Y., Xie, W., 2017. Nonanticipative duality, relaxations, and formulations for chance-constrained stochastic programs. *Mathematical Programming* 162, 51–81.
- Alamo, T., Tempo, R., Luque, A., Ramirez, D.R., 2015. Randomized methods for design of uncertain systems: Sample complexity and sequential algorithms. *Automatica* 52, 160–172.
- Bandi, C., Bertsimas, D., 2012. Tractable stochastic analysis in high dimensions via robust optimization. *Mathematical Programming* 134, 23–70.
- Basak, S., Shapiro, A., 2001. Value-at-risk-based risk management: Optimal policies and asset prices. *The Review of Financial Studies* 14, 371–405.
- Ben-Tal, A., Bertsimas, D., Brown, D.B., 2010. A soft robust model for optimization under ambiguity. *Operations Research* 58, 1220–1234.
- Ben-Tal, A., Brekelmans, R., Den Hertog, D., Vial, J.P., 2017. Globalized robust optimization for nonlinear uncertain inequalities. *INFORMS Journal on Computing* 29, 350–366.
- Ben-Tal, A., El Ghaoui, L., Nemirovski, A., 2009. *Robust Optimization*. Princeton university press.
- Ben-Tal, A., den Hertog, D., de Waegenare, A., Melenberg, B., Rennen, G., 2013. Robust solutions of optimization problems affected by uncertain probabilities. *Management Science* 59, 341–357.
- Bertsimas, D., Brown, D.B., Caramanis, C., 2011. Theory and applications of robust optimization. *SIAM Review* 53, 464–501.
- Bertsimas, D., Gupta, V., Kallus, N., 2018. Data-driven robust optimization. *Mathematical Programming* 167, 235–292.
- Bertsimas, D., den Hertog, D., 2022. *Robust and Adaptive Optimization*. Dynamic Ideas LLC.
- Bertsimas, D., den Hertog, D., Pauphilet, J., 2021. Probabilistic guarantees in robust optimization. *SIAM Journal on Optimization* 31, 2893–2920.

- Bertsimas, D., den Hertog, D., Pauphilet, J., Zhen, J., 2023. Robust convex optimization: A new perspective that unifies and extends. *Mathematical Programming* 200, 877–918.
- Bertsimas, D., de Ruiter, F., 2016. Duality in two-stage adaptive linear optimization: Faster computation and stronger bounds. *INFORMS Journal on Computing* 28, 500–511.
- Birge, J.R., 1997. State-of-the-art-survey – Stochastic programming: Computation and applications. *INFORMS Journal on Computing* 9, 111–133.
- Birge, J.R., Louveaux, F., 2011. *Introduction to Stochastic Programming*. Springer Science & Business Media.
- Calafiore, G.C., 2013. Direct data-driven portfolio optimization with guaranteed shortfall probability. *Automatica* 49, 370–380.
- Calafiore, G.C., 2017. Repetitive scenario design. *IEEE Transactions on Automatic Control* 62, 1125–1137.
- Calafiore, G.C., Campi, M.C., 2005. Uncertain convex programs: Randomized solutions and confidence levels. *Mathematical Programming* 102, 25–46.
- Calafiore, G.C., Ghaoui, L.E., 2006. On distributionally robust chance-constrained linear programs. *Journal of Optimization Theory and Applications* 130, 1–22.
- Campi, M.C., Garatti, S., 2008. The exact feasibility of randomized solutions of uncertain convex programs. *SIAM Journal on Optimization* 19, 1211–1230.
- Campi, M.C., Garatti, S., 2018. Wait-and-judge scenario optimization. *Mathematical Programming* 167, 155–189.
- Carè, A., Garatti, S., Campi, M.C., 2014. FAST—fast algorithm for the scenario technique. *Operations Research* 62, 662–671.
- Chamanbaz, M., Dabbene, F., Tempo, R., Venkataramanan, V., Wang, Q.G., 2015. Sequential randomized algorithms for convex optimization in the presence of uncertainty. *IEEE Transactions on Automatic Control* 61, 2565–2571.
- Charnes, A., Cooper, W., 1959. Chance-constrained programming. *Management Science* 6, 73–79.
- Delage, E., Ye, Y., 2010. Distributionally robust optimization under moment uncertainty with application to data-driven problems. *Operations Research* 58, 595–612.
- Fischetti, M., Monaci, M., 2009. Light robustness. *Robust and Online Large-Scale Optimization: Models and Techniques for Transportation Systems*, 61–84.
- Garatti, S., Carè, A., Campi, M.C., 2022. Complexity is an effective observable to tune early stopping in scenario optimization. *IEEE Transactions on Automatic Control* 68, 928–942.
- Guzman, Y.A., Matthews, L.R., Floudas, C.A., 2016. New a priori and a posteriori probabilistic bounds for robust counterpart optimization: I. unknown probability distributions. *Computers & Chemical Engineering* 84, 568–598.

- Hanasusanto, G.A., Roitch, V., Kuhn, D., Wiesemann, W., 2015. A distributionally robust perspective on uncertainty quantification and chance constrained programming. *Mathematical Programming* 151, 35–62.
- Hong, L.J., Huang, Z., Lam, H., 2021. Learning-based robust optimization: Procedures and statistical guarantees. *Management Science* 67, 3447–3467.
- Jiang, N., Xie, W., 2022. ALSO-X and ALSO-X+: Better convex approximations for chance constrained programs. *Operations Research* 70, 3581–3600.
- Jiang, R., Guan, Y., 2016. Data-driven chance constrained stochastic program. *Mathematical Programming* 158, 291–327.
- Laeven, R.J., Stadje, M., 2014. Robust portfolio choice and indifference valuation. *Mathematics of Operations Research* 39, 1109–1141.
- Luedtke, J., 2014. A branch-and-cut decomposition algorithm for solving chance-constrained mathematical programs with finite support. *Mathematical Programming* 146, 219–244.
- Luedtke, J., Ahmed, S., 2008. A sample approximation approach for optimization with probabilistic constraints. *SIAM Journal on Optimization* 19, 674–699.
- Luedtke, J., Ahmed, S., Nemhauser, G.L., 2010. An integer programming approach for linear programs with probabilistic constraints. *Mathematical Programming* 122, 247–272.
- Mohajerin Esfahani, P., Kuhn, D., 2018. Data-driven distributionally robust optimization using the wasserstein metric: Performance guarantees and tractable reformulations. *Mathematical Programming* 171, 115–166.
- Natarajan, K., Pachamanova, D., Sim, M., 2008. Incorporating asymmetric distributional information in robust value-at-risk optimization. *Management Science* 54, 573–585.
- Nemirovski, A., Shapiro, A., 2006. Scenario approximations of chance constraints. *Probabilistic and Randomized Methods for Design under Uncertainty*, 3–47.
- Nemirovski, A., Shapiro, A., 2007. Convex approximations of chance constrained programs. *SIAM Journal of Optimization* 17, 969–996.
- Norris, J., 1997. *Markov Chains*. Cambridge University Press.
- Oishi, Y., 2007. Polynomial-time algorithms for probabilistic solutions of parameter-dependent linear matrix inequalities. *Automatica* 43, 538–545.
- Pagnoncelli, B.K., Ahmed, S., Shapiro, A., 2009. Sample average approximation method for chance constrained programming: Theory and applications. *Journal of Optimization Theory and Applications* 142, 399–416.
- Pardo, L., 2006. *Statistical Inference Based on Divergence Measures*. Chapman & Hall/CRC Boca Raton.

- Postek, K., Ben-Tal, A., Den Hertog, D., Melenberg, B., 2018. Robust optimization with ambiguous stochastic constraints under mean and dispersion information. *Operations Research* 66, 814–833.
- Postek, K., den Hertog, D., Melenberg, B., 2016. Computationally tractable counterparts of distributionally robust constraints on risk measures. *SIAM Review* 58, 603–650.
- Rahimian, H., Mehrotra, S., 2019. Frameworks and results in distributionally robust optimization. *Open Journal of Mathematical Optimization* 3, 1–85.
- Roos, E., den Hertog, D., 2020. Reducing conservatism in robust optimization. *INFORMS Journal on Computing* 32, 1109–1127.
- Shang, C., You, F., 2020. A posteriori probabilistic bounds of convex scenario programs with validation tests. *IEEE Transactions on Automatic Control* 66, 4015–4028.
- Shapiro, A., Dentcheva, D., Ruszczyński, A., 2009. *Lectures On Stochastic Programming*. SIAM.
- Shapiro, A., Nemirovski, A., 2005. On complexity of stochastic programming problems. *Continuous Optimization: Current Trends and Modern Applications* , 111–146.
- Shawe-Taylor, J., Cristianini, N., 2003. Estimating the moments of a random vector with applications. *Proceedings of GRETSI 2003 Conference* , 47–52.
- Smith, R.L., 1984. Efficient Monte Carlo procedures for generating points uniformly distributed over bounded regions. *Operations Research* 32, 1296–1308.
- Vayanos, P., Kuhn, D., Rustem, B., 2012. A constraint sampling approach for multi-stage robust optimization. *Automatica* 48, 459–471.
- Wallace, S.W., Ziemba, W.T., 2005. *Applications of Stochastic Programming*. SIAM.
- Wang, I., Becker, C., Van Parys, B., Stellato, B., 2024. Mean robust optimization. *Mathematical Programming* , 1–43.
- Yanıkoglu, İ., den Hertog, D., 2013. Safe approximations of ambiguous chance constraints using historical data. *INFORMS Journal on Computing* 25, 666–681.