# ROBIST: Robust Optimization by Iterative Scenario Sampling and Statistical Testing

Justin Starreveld
Department of Business Analytics, University of Amsterdam, Amsterdam, 1001 NL, Netherlands, j.s.starreveld@uva.nl

Guanyu Jin
Departement of Quantitative Economics, University of Amsterdam, Amsterdam, 1001 NL, Netherlands, g.jin@uva.nl

Dick den Hertog
Department of Business Analytics, University of Amsterdam, Amsterdam, 1001 NL, Netherlands, d.denhertog@uva.nl

Roger J. A. Laeven
Department of Quantitative Economics, University of Amsterdam, Amsterdam, 1001 NL, Netherlands, r.j.a.laeven@uva.nl

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.

*Key words*: optimization; parametric uncertainty; robust; data-driven

## 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.[1] 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:

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

[1] This observation is made on the basis of anecdotal evidence obtained through extensive contact with industry.

where $\mathbf{x} \in \mathbb{R}^{d_\mathbf{x}}$ is a decision vector, restricted to a closed convex feasible set $\mathscr{X}$, $\mathbf{z} \in \mathbb{R}^{d_\mathbf{z}}$ is an uncertain parameter vector and $g(\mathbf{x})$ and $f(\mathbf{z}, \mathbf{x})$ are scalar-valued functions that are 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 the uncertainty to the constraints by using an epigraph formulation. Furthermore, note that multiple constraints can be incorporated into a single constraint by defining $f(\mathbf{z}, \mathbf{x}) := \max_{j=1,\ldots,m} \{f_j(\mathbf{z}, \mathbf{x})\} \leq 0$, for $m$ individual constraints $f_j(\mathbf{z}, \mathbf{x}) \leq 0$.

The problem (UCP) as formulated above is not well-defined as it does not specify how the uncertain constraint $f(\mathbf{z}, \mathbf{x}) \leq 0$ 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 $\mathbf{z}$.

## 1.1. Existing Approaches and Their Practical Limitations

In the following paragraphs we describe four well-known approaches for treating uncertain constraints and highlight limitations to their application in practice.

**1.1.1. Stochastic Programming.** Stochastic programming (SP) assumes that the uncertain parameters are of a stochastic nature, i.e., that $\mathbf{z}$ is a realization of the random variable $\tilde{\mathbf{z}}$ with some known probability distribution $\mathbb{P}$. For a more elaborate description of SP, we refer to Birge and Louveaux (2011). In the context of this paper, where we assume to be interested in obtaining a robust solution to (UCP), one may consider the "chance-constrained" programming approach. This approach was first proposed by Charnes and Cooper (1959), where the uncertain constraint $f(\mathbf{z}, \mathbf{x}) \leq 0$ is replaced by a probabilistic constraint:

$$\mathbb{P}(f(\tilde{\mathbf{z}}, \mathbf{x}) \leq 0) \geq 1 - \epsilon, \tag{1}$$

for some acceptable probability of constraint violation $\epsilon \geq 0$.

While SP has been successfully applied to small- and medium-sized problems (Birge 1997, Wallace and Ziemba 2005), there are limitations to its application in practice. First, the probability distribution $\mathbb{P}$ is often unknown. Second, even if $\mathbb{P}$ is known, exact tractable reformulations of (1) are only known for a limited number of situations (Shapiro and Nemirovski 2005). Third, the computational tractability of SP deteriorates as $d_\mathbf{z}$ increases (Nemirovski et al. 2009). As such, this approach is considered generally intractable and one is typically forced to resort to safe approximations of (1) when dealing with large-scale problems (Nemirovski 2012).

**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 $f(\mathbf{z},\mathbf{x}) \leq 0$ is thus formalized as:

$$f(\mathbf{z},\mathbf{x}) \leq 0, \quad \forall \mathbf{z} \in \mathcal{U}. \tag{2}$$

This approach has emerged as a tractable alternative to SP for high-dimensional problems, see, for example, Bandi and Bertsimas (2012). Furthermore, in certain cases, (2) can be constructed to be equivalent to probabilistic constraints such as (1), see e.g., Boyd and Vandenberghe (2004, pp. 157–158). For an 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 $\mathcal{U}$, which may pose a hindrance to its implementation in practice. Aside from selecting an appropriate shape for $\mathcal{U}$, it can also be difficult to select an appropriate size. If the size of the uncertainty set is too small, then the resulting solution may not be sufficiently robust. On the contrary, if the size is too large, the resulting solution may be overly conservative. A variety of methods have been proposed to eliminate this hindrance by utilizing data to determine $\mathcal{U}$ in such a way that one can have a certain level of confidence that any solution $\mathbf{x}$ satisfying (2) also satisfies (1). Two such methods are proposed by Yanıkoğlu and den Hertog (2013) and Bertsimas et al. (2018), however, as we demonstrate via numerical experiments in this paper, the former method scales poorly in $d_{\mathbf{z}}$ and the latter method may nevertheless result in overly conservative solutions.

The tendency for classic RO methods to result in overly conservative solutions 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 any case, RO relies on the ability to reformulate (2) to a tractable robust counterpart, which is not always possible in practice. First, if $f$ is concave in $\mathbf{z}$, even though the resulting robust counterpart from reformulating (2) may theoretically be solvable in polynomial time, in practice it may involve complex nonlinear constraints and/or an unacceptably large number of additional variables and constraints (Bertsimas et al. 2011). Second, if the function $f$ is non-concave in $\mathbf{z}$, exact reformulations of (2) 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 the application in practice.

**1.1.3. Distributionally robust optimization.** Also referred to as "ambiguous stochastic programming", distributionally robust optimization (DRO), combines elements of SP and RO. Instead of assuming the probability distribution $\mathbb{P}$ of $\tilde{\mathbf{z}}$ is known (as in SP), the distribution is regarded as uncertain, yet restricted to an "ambiguity set" $\mathcal{P}$ of possible distributions. The uncertain constraint can then be formulated as:

$$\mathbb{P}(f(\tilde{\mathbf{z}},\mathbf{x}) \le 0) \ge 1-\epsilon, \quad \forall \mathbb{P} \in \mathcal{P}. \tag{3}$$

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 (3) we refer to Postek et al. (2018).

Similarly to RO, the key issue 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. 2022).

The ability to reformulate (3) 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 (3) 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 technique within RO, where the uncertainty set $\mathcal{U}$ is a finite set of "scenarios" $\mathcal{S}$. Because the set $\mathcal{S}$ is finite, there is no need to reformulate (2) and the uncertain constraint is simply replicated for each scenario $\mathbf{z}^j$ in the set $\mathcal{S}$. This amounts to solving a finite scenario convex program (SCP):

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

The SO approach is easy 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 $\mathbf{x}$ 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), the method proposed by Calafiore and Campi (2005) 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 $d_{\mathbf{x}}$ (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 (2016), 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 to (UCP). 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 method of Yanıkoğlu and den Hertog (2013). However, our evaluation procedure for assessing the robustness of solutions differs from theirs in significant ways, which is discussed in more detail in Section 2. 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.

The novelty in our approach is that the statistical tests are carried out on the *univariate* random variable $f(\tilde{\mathbf{z}}, \mathbf{x})$, where $\mathbf{x}$ is fixed. As we are concerned with the feasibility of $\mathbf{x}$, there are only two "classes" (i.e., $f(\tilde{\mathbf{z}}, \mathbf{x}) \leq 0$ and $f(\tilde{\mathbf{z}}, \mathbf{x}) > 0$). This provides sharp probability guarantees that are independent of the number of decision variables as well as the number of 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 method offers a number of practical advantages over existing methods in the literature. In summary, the advantages of ROBIST are the following.

First, our method is highly versatile, 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.

Second, our method is more accessible than many existing methods within SP, RO and DRO. It is data-driven and does not require any assumptions regarding the underlying probability distribution of $\tilde{\mathbf{z}}$. Additionally, our method avoids the aforementioned difficulties in regard to selecting an appropriate uncertainty or ambiguity set.

Third, 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 novel 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.

Fourth, our method is able to explore the trade-off between optimality and robustness and can offer insight into the "price of robustness" (Bertsimas and Sim 2004). The user sets a desired level of robustness and, with the aid of sharp probability guarantees, our method is able to identify and avoid overly conservative solutions.

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 versatility and effectiveness of the algorithm via a variety of numerical experiments. Finally, we provide concluding remarks in Section 5. Additional technical details and proofs 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:

$$\mathbb{P}(f(\tilde{\mathbf{z}}, \mathbf{x}) \le 0) \ge \gamma. \tag{4}$$

Henceforth, we will refer to $\gamma$ as a *feasibility certificate*. Note that if a solution $\mathbf{x}$ satisfies (4) with $\gamma \ge 1-\epsilon$, one can state, with a certain level of confidence, that $\mathbf{x}$ satisfies (1). This insight serves as the foundation to our proposed algorithm for treating (UCP).

### 2.1. Algorithm

Our algorithm, ROBIST, consists of two main procedures: (i) a generation procedure in which we use a training data set $\mathcal{D}^{\text{train}}_{N_1} = \left\{\hat{\mathbf{z}}^1, \dots, \hat{\mathbf{z}}^{N_1}\right\}$ to generate solutions, and (ii) an evaluation procedure in which we use a testing data set $\mathcal{D}^{\text{test}}_{N_2} = \left\{\check{\mathbf{z}}^1, \dots, \check{\mathbf{z}}^{N_2}\right\}$ to evaluate the robustness of the generated solutions. By embedding these two procedures in an iterative algorithm, we look to obtain solutions $\mathbf{x}$ that satisfy (4) with $\gamma \ge 1-\epsilon$, 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 $\mathbf{x}_i$:

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

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), our algorithm is designed to keep the size of $\mathcal{S}_i$ to a minimum.

**2.1.2. Evaluation procedure.** Given a data set $\mathcal{D}_N = \{\mathbf{z}^1, \ldots, \mathbf{z}^N\}$ and some solution $\mathbf{x}_i$, one can evaluate the robustness of $\mathbf{x}_i$ using $\mathcal{D}_N$ and derive a feasibility certificate $\gamma_i$ via statistical testing. Our procedure is as follows. First, for each scenario $\mathbf{z}^j \in \mathcal{D}_N$, we compute $f(\mathbf{z}^j, \mathbf{x}_i)$ and check whether $f(\mathbf{z}^j, \mathbf{x}_i) \leq 0$ is satisfied. This provides an empirical estimate $p$ of the probability that $\mathbf{x}_i$ is feasible:

$$p := \frac{1}{N} \sum_{j=1}^{N} \mathbb{1}_{[f(\mathbf{z}^j, \mathbf{x}_i) \leq 0]}. \tag{5}$$

Second, we construct a statistical confidence interval around (5) using the modified $\chi^2$-distance, which is a member of the family of *$\phi$-divergences* (see Appendix A for further details). This results in the following $1-\alpha$ confidence region:

$$\mathcal{Q}_\phi(p, N, \alpha) := \left\{ q \in \mathbb{R} \ : \ q \geq 0, \ \frac{(q-p)^2}{p} + \frac{(q-p)^2}{1-p} \leq \frac{\chi^2_{1,1-\alpha}}{N} \right\}, \tag{6}$$

where $\alpha$ is determined by the user and $\chi^2_{1,1-\alpha}$ is the $1-\alpha$ quantile of the chi-squared distribution with 1 degree of freedom. As shown by Pardo (2006), as $N \to \infty$, the set (6) contains the true probability that $\mathbf{x}_i$ is feasible with confidence of at least $1-\alpha$.

Third, we determine feasibility certificate $\gamma_i$, which implies a probabilistic guarantee equivalent to (4) for solution $\mathbf{x}_i$, by computing:

$$\gamma_i := \min_{q \in \mathcal{Q}_\phi(p, N, \alpha)} q. \tag{7}$$

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

Lemma 1. *The problem stated in* (7) *has the following closed-form solution:*

$$\min_{q \in \mathcal{Q}_\phi(p, N, \alpha)} q = \max\left\{ p - \sqrt{p(1-p)r}, 0 \right\}, \quad \textit{with } r = \frac{\chi^2_{1,1-\alpha}}{N}. \tag{8}$$

*Furthermore, $\gamma_i$ is an increasing function in $p$.*

Note that certificates derived via (7) are only statistically valid if the scenarios $\mathbf{z}^j \in \mathcal{D}_N$ are independent realizations of $\tilde{\mathbf{z}}$. This is why ROBIST utilizes a separate testing data set $\mathcal{D}_{N_2}^{\text{test}}$ to derive a valid certificate $\gamma_i$ for each generated solution $\mathbf{x}_i$.

The use of $\phi$-divergence for constructing uncertainty sets has appeared in existing robust optimization literature, we refer to Ben-Tal et al. (2013) for an overview. However, in these methods the confidence region is constructed around the empirical distribution of $\tilde{\mathbf{z}}$. The degrees of freedom in the resulting $\phi$-divergence confidence set is then dependent on $d_{\mathbf{z}}$, which may lead to overly conservative uncertainty sets when dealing with problems with a large number of uncertain parameters. Our approach circumvents this issue by constructing the set around (5), which is independent of $d_{\mathbf{z}}$

**2.1.3. The scenario sets.** 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 $\mathcal{D}_{N_2}^{\text{test}}$. The idea is that $\hat{\gamma}_i$, while statistically invalid, can act as an estimate of $\gamma_i$ 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 $\upsilon$, the opposite action is taken. This random component is added to ensure theoretical convergence of our algorithm, which we discuss in Section 2.3.

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 $\{\hat{\mathbf{z}}^j \in \mathcal{D}_{N_1}^{\text{train}} : f(\hat{\mathbf{z}}^j, \mathbf{x}_i) > 0\}$. 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 $f(\hat{\mathbf{z}}^j, \mathbf{x}) \leq 0$ 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 $\gamma_{i+1} = \gamma_i$.

**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 minimal objective value $g(\mathbf{x}_i)$, while requiring that $\gamma_i \geq 1-\epsilon$.

**Algorithm 1** ROBIST

```
Input: Sets $\mathcal{D}_{N_1}^{\text{train}} = \{\hat{\mathbf{z}}^1, \dots, \hat{\mathbf{z}}^{N_1}\}$ and $\mathcal{D}_{N_2}^{\text{test}} = \{\check{\mathbf{z}}^1, \dots, \check{\mathbf{z}}^{N_2}\}$. Acceptable probability
    of constraint violation $\epsilon$, statistical significance level $\alpha$, time limit $T_{\max}$, iteration
    limit $i_{\max}$ and probability of taking opposite action $\upsilon$.
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 $T < T_{\max}$ and $i < i_{\max}$ do
 5:     Solve (SCP) to obtain $\mathbf{x}_i$
 6:     Derive proxy certificate $\hat{\gamma}_i$ via (8) 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 $\iota > \upsilon$) or ($\hat{\gamma}_i > 1-\epsilon$ and $\iota < \upsilon$) then
 9:         Randomly add a scenario from $\{\hat{\mathbf{z}}^j \in \mathcal{D}_{N_1}^{\text{train}} : f(\hat{\mathbf{z}}^j, \mathbf{x}_i) > 0\}$ 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 feasibility certificates $\gamma_j, j = 0, \dots, i-1$ via (8) using the scenarios in $\mathcal{D}_{N_2}^{\text{test}}$
16: if $\exists \gamma_j : \gamma_j \geq 1-\epsilon$ then
17:     $i^* := \operatorname{argmin}_j \{g(\mathbf{x}_j) : \gamma_j \geq 1-\epsilon\}$
18: else
19:     $i^* := \operatorname{argmax}_j \{\gamma_j\}$
20: end if
21: Return $\mathbf{x}_{i^*}$
```

### 2.2. Illustrative Example

Consider the following toy problem from Yanıkoğlu and den Hertog (2013):

$$\max_{x_1, x_2 \leq 1} \; x_1 + x_2 \tag{9}$$

$$\text{s.t. } z_1 x_1 + z_2 x_2 \leq 1, \tag{10}$$

where $z_1$ and $z_2$ are uncertain parameters, both uniformly distributed with support $[-1,1]$. Note that Problem (9)-(10) can be rewritten in the same form as (UCP) by defining: $\mathscr{X} = \{\mathbf{x} : x_1 \leq 1, x_2 \leq 1\}, g(\mathbf{x}) = -(x_1 + x_2)$ and $f(\mathbf{z}, \mathbf{x}) = z_1 x_1 + z_2 x_2 - 1$.

Imagine we have access to a data set of $N = 200$ realizations of $(\tilde{z}_1, \tilde{z}_2)$ and would like a solution to be feasible with probability of at least 90% (i.e., $\epsilon = 0.1$). In the following paragraphs we illustrate the application of our method to this problem.

First, we randomly split the data set into two equal-sized sets $\mathcal{D}_{N_1}^{\text{train}} = \{\hat{\mathbf{z}}^1, \dots, \hat{\mathbf{z}}^{N_1}\}$ and $\mathcal{D}_{N_2}^{\text{test}} = \{\check{\mathbf{z}}^1, \dots, \check{\mathbf{z}}^{N_2}\}$, each containing $N_1 = N_2 = 100$ 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, towards the end of the algorithm, we utilize $\mathcal{D}_{N_2}^{\text{test}}$ to derive statistically valid probability guarantees.

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) 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 (10) by determining whether $f(\hat{\mathbf{z}}^j, \mathbf{x}_0) \le 0,\ \forall \hat{\mathbf{z}}^j \in \mathcal{D}_{N_1}^{\text{train}}$. See Figure 1 for a visual aid.

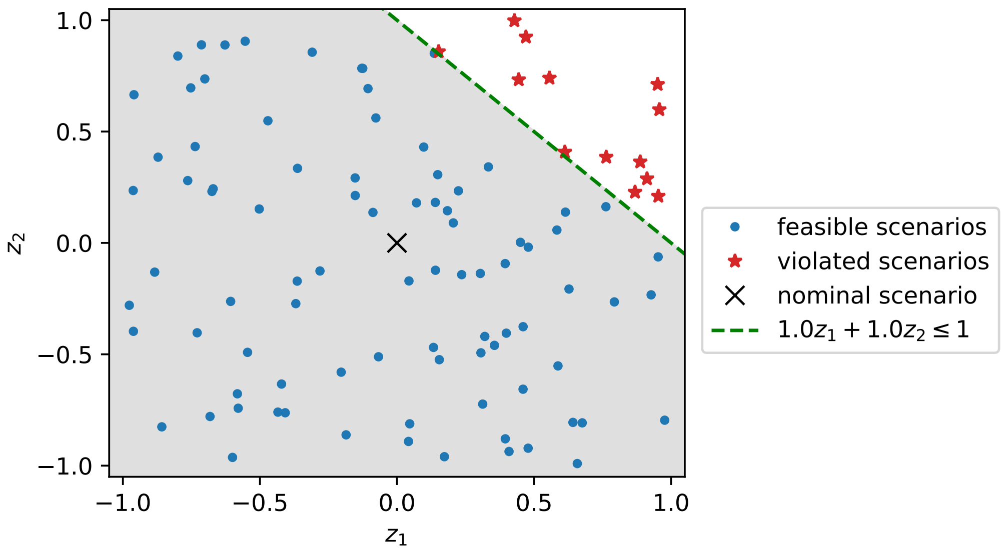


**Figure 1** **Visualization of $\mathcal{S}_0$ and the evaluation of the constraint $f(\mathbf{x}_0, \hat{\mathbf{z}}^j) \le 0$ of the solution $\mathbf{x}_0 = (1, 1)$ on the training data $\mathcal{D}_{N_1}^{\text{train}}$. The data points for which the constraint is feasible/infeasible are indicated 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 (8), 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) 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 $\mathcal{D}^{\text{train}}_{N_1}$ (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 $\hat{\gamma}_1=0.93$.

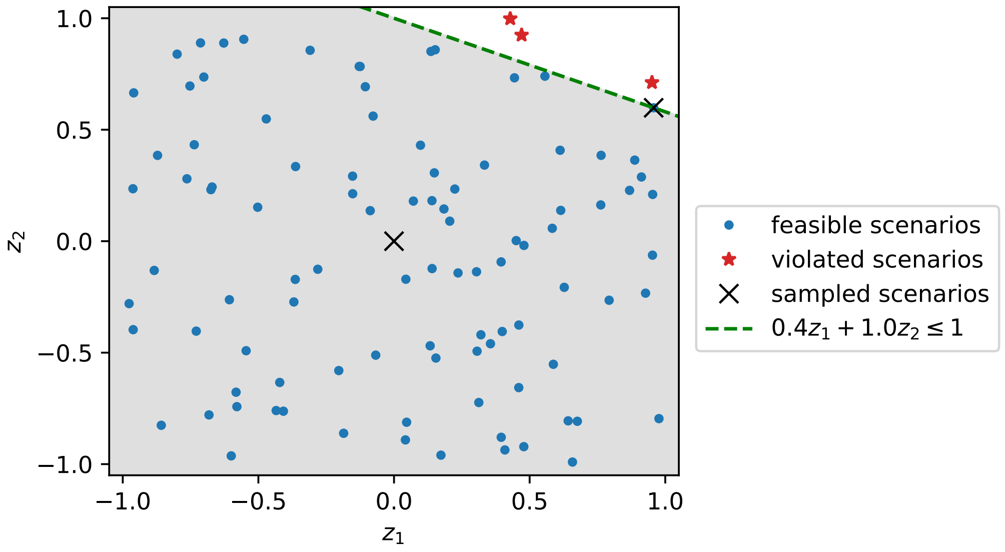


**Figure 2** **Visualization of $\mathcal{S}_1$ and the evaluation of solution $\mathbf{x}_1=(0.4,1)$ on the training data.**

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 $\mathcal{D}^{\text{train}}_{N_1}$ in this manner until either the time limit or iteration limit is reached.

In this example, we set a limit of 1,000 iterations and once this stopping criteria is reached, we use the "out-of-sample" test data $\mathcal{D}^{\text{test}}_{N_2}$ to properly evaluate each solution $\mathbf{x}_i$ and obtain valid feasibility certificates $\gamma_i$. These evaluations can then be used to construct a trade-off curve and aid in choosing the "best" solution. The blue line in Figure 3 depicts such a trade-off curve, where each blue circle represents a possible solution. If one considers $\gamma_i\geq 0.90$ to be a strict requirement, the best found solution comes with a feasibility certificate of 0.910 and achieves an objective value of 1.32. However, in this example we find a sharp drop off after 0.90 — a slightly less conservative solution was found with a certificate of 0.894 and an objective value of 1.50.

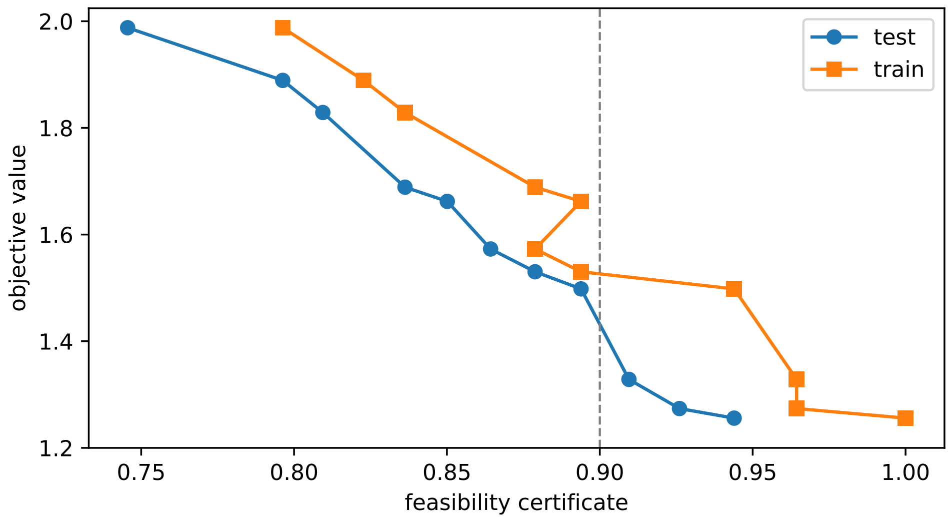


**Figure 3** **Trade-off curves constructed from the set of non-dominated solutions found by ROBIST. The blue circles represent valid feasibility certificates derived using the test data, while the orange squares represent the associated proxy certificates derived using the training data. The vertical dotted line represents the user-defined desired level of robustness $(1-\epsilon)$.**

## 2.3. Convergence

In this section, we investigate the optimality of the final solution generated by Algorithm 1. Since the solutions are generated by solving (SCP) with respect to a subset $\mathcal{S}_i$ of the training data set, the natural question arises whether Algorithm 1 would eventually yield an "optimal" solution with respect to the training data set.

We define a solution $\mathbf{x}$ to be optimal with respect to the training data $\mathcal{D}_{N_1}^{\text{train}} = \{\hat{\mathbf{z}}^1, \ldots, \hat{\mathbf{z}}^{N_1}\}$ if the following conditions are satisfied:

- $\mathbf{x}$ is a solution obtained from solving Problem (SCP$_{\mathcal{S}^*}$) with respect to a specific subset $\mathcal{S}^*$ of $\mathcal{D}_{N_1}^{\text{train}}$,
- $\mathbf{x}$ is a solution with a valid feasibility certificate $\gamma \geq 1-\epsilon$ (with respect to the test data $\mathcal{D}_{N_2}^{\text{test}}$),
- $\mathbf{x}$ has minimal objective $g(\mathbf{x})$ compared to all other solutions (with $\gamma \geq 1-\epsilon$) obtained from solving (SCP) with respect to a subset $\mathcal{S}_i$ of $\mathcal{D}_{N_1}^{\text{train}}$.

The following lemma shows that Algorithm 1 yields a solution that is optimal with respect to the training 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 $\upsilon > 0$. Suppose that for all possible subsets $\mathcal{S}_i \subseteq \mathcal{D}_{N_1}^{\text{train}}$, the solution to* (SCP) *is unique, and that there exists a* (SCP) *such that its solution has a feasibility certificate* (7) *of at least $1-\epsilon$, with respect to the test data $\mathcal{D}_{N_2}^{\text{test}}$. Then, the probability that the solution*

$\mathbf{x}^*$ *obtained from Algorithm 1 after finitely many iterations, is optimal with respect to* $\mathcal{D}_{N_1}^{\text{train}}$*, is 1.*

REMARK 1. The assumption used in the proof of Lemma 2 on the uniqueness of solutions of all (SCP) is unnecessary if the solver that is used to solve (SCP) 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:

$$\min_{\mathbf{x}\in\mathscr{X}} f(\mathbf{z},\mathbf{x}). \tag{11}$$

Problem (11) can be reformulated to the same form as (UCP) by using an epigraph formulation, which results in the following:

$$\min_{\mathbf{x},\theta} \theta \tag{12}$$

$$\text{s.t. } f(\mathbf{z},\mathbf{x}) \le \theta, \tag{13}$$

$$\mathbf{x}\in\mathscr{X}. \tag{14}$$

Note that the right-hand side of (13) is determined by the epigraph *variable* $\theta \in \mathbb{R}$, which can always be adjusted such that (13) is satisfied. As such, whereas (11) can be rewritten in the same form as (UCP), the uncertain "constraint" should be treated differently.

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

$$\mathbb{P}(f(\tilde{\mathbf{z}},\mathbf{x}_i)\le\theta_i)\ge\gamma_i, \tag{15}$$

we are now interested in determining the minimum value of $\theta_i$, let this be denoted by $\theta_i^{\min}$, for which we can claim, with confidence of at least $1-\alpha$, that:

$$\mathbb{P}(f(\tilde{\mathbf{z}},\mathbf{x}_i)\le\theta_i^{\min})\ge 1-\epsilon. \tag{16}$$

We determine $\theta_i^{\min}$ in the following manner. First, we determine (via golden-section search) the threshold for the number of scenarios in $\mathcal{D}_{N_2}^{\text{test}}$ for which the "constraint" (13) must be satisfied in order to claim (16). Denote this threshold by $N_2^{\min}$. Then, we sort function evaluations $\{f(\mathbf{z}^j, \mathbf{x}_i), \forall \mathbf{z}^j \in \mathcal{D}_{N_2}^{\text{test}}\}$ and set $\theta_i^{\min}$ equal to the $N_2^{\min}$-th largest evaluation.

Finally, since there are no uncertain constraints in (11), there is no trade-off between feasibility and optimality. In such situations, the problem becomes a one-dimensional search for the minimal $\theta_i^{\min}$ for which we can claim (16). As such, Steps 16-20 in Algorithm 1 are also adjusted, where we now define $i^* := \operatorname{argmin}_j\{\theta_j^{\min}\}$.

## 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:

$$\min_{\mathbf{x}\in\mathscr{X}} \; g(\mathbf{x}) \tag{17}$$

$$\text{s.t. } V(\mathbf{z},\mathbf{x}) \le 0, \tag{18}$$

where

$$V(\mathbf{z},\mathbf{x}) := \min_{\mathbf{y}\in\mathscr{Y}(\mathbf{x})} f(\mathbf{z},\mathbf{x},\mathbf{y}). \tag{19}$$

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 $\mathscr{Y}(\mathbf{x})$, which may depend on the first-stage decisions $\mathbf{x}$.

We are still able to generate solutions to Problem (17)-(19) 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:

$$\min_{\mathbf{x},\mathbf{y}} \; g(\mathbf{x}) \tag{20}$$

$$\text{s.t. } f(\hat{\mathbf{z}}^j, \mathbf{x}, \mathbf{y}^j) \le 0, \qquad \forall\, \hat{\mathbf{z}}^j \in \mathcal{S}_i, \tag{21}$$

$$\mathbf{y}^j \in \mathscr{Y}(\mathbf{x}), \qquad \forall\, \hat{\mathbf{z}}^j \in \mathcal{S}_i, \tag{22}$$

$$\mathbf{x} \in \mathscr{X}. \tag{23}$$

By solving (20)-(23) 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 $f(\mathbf{z}^j,\mathbf{x}_i)$ for $j=1,\dots,N$), one now evaluates $V(\mathbf{z}^j,\mathbf{x}_i), j=1,\dots,N$ by solving $N$ instances of Problem (19). This allows one to determine whether there exists a recourse decision $\mathbf{y}$ such that uncertain constraint (18) could be satisfied. With this information one can compute empirical estimate $p$, where:

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

This estimate $p$ can then be used to derive feasibility certificate $\gamma_i$ by computing (7).

### 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 as in (1). 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:

$$\mathbb{E}_{\mathbb{P}}[f(\tilde{\mathbf{z}},\mathbf{x})], \tag{24}$$

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 $f(\tilde{\mathbf{z}},\mathbf{x}_i)$ and denote by $\mathbb{P}_{\mathbf{x}_i}$ the distribution of $f(\tilde{\mathbf{z}},\mathbf{x}_i)$ on $[l_{\mathbf{x}_i},u_{\mathbf{x}_i}]$. One can construct a partition $\{[e_k,e_{k+1}]\}_{k=1}^K$, where $l_{\mathbf{x}_i}=e_1\le e_2\le\cdots\le e_{K+1}=u_{\mathbf{x}_i}$. Then, the following inequality must hold:

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

Let vector $\mathbf{p}\in\mathbb{R}^K$ be an empirical estimate of the probability (under $\mathbb{P}_{\mathbf{x}_i}$) that $f(\tilde{\mathbf{z}},\mathbf{x}_i)$ 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:

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

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

$$\mathcal{Q}_\phi(\mathbf{p},N,\alpha) = \left\{\mathbf{q}\in\mathbb{R}^K : \mathbf{q}\ge\mathbf{0}, \mathbf{e}^\intercal\mathbf{q}=1, I_\phi(\mathbf{q},\mathbf{p})\le\frac{\phi''(1)}{2N}\chi^2_{K-1,1-\alpha}\right\}. \tag{25}$$

As $N \to \infty$, the set (25) contains the true probability that $f(\tilde{\mathbf{z}}, \mathbf{x}_i)$ resides in each of the $K$ intervals. As such, one can get an asymptotic upper confidence bound $u_i$ on $\mathbb{E}_{\mathbb{P}}[f(\tilde{\mathbf{z}}, \mathbf{x}_i)]$, by computing:

$$u_i := \max_{\mathbf{q} \in \mathcal{Q}_\phi(\mathbf{p}, N, \alpha)} \sum_{k=1}^{K} e_{k+1} q_k. \tag{26}$$

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

$$l_i := \min_{\mathbf{q} \in \mathcal{Q}_\phi(\mathbf{p}, N, \alpha)} \sum_{k=1}^{K} e_k q_k. \tag{27}$$

The optimization problems stated in (26) and (27) 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 $R(\mathbf{z}^j, \mathbf{x}_i)$ with respect to a realized scenario $\mathbf{z}^j$, as:

$$R(\mathbf{z}^j, \mathbf{x}_i) := \begin{cases} g(\mathbf{x}_i) - g^*(\mathbf{z}^j), & \text{if } f(\mathbf{z}^j, \mathbf{x}_i) \le 0, \\ +\infty, & \text{otherwise,} \end{cases} \tag{28}$$

where:

$$g^*(\mathbf{z}^j) := \min_{\mathbf{x} \in \mathcal{X}} \{g(\mathbf{x}) : f(\mathbf{z}^j, \mathbf{x}) \le 0\}. \tag{29}$$

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 $f(\mathbf{z}^j, \mathbf{x}_i)$, we evaluate $R(\mathbf{z}^j, \mathbf{x}_i)$, as defined in (28). These evaluations can then be used to claim, with confidence of at least $1 - \alpha$, that:

$$\mathbb{P}(R(\tilde{\mathbf{z}}, \mathbf{x}_i) \le \tau) \ge \beta_i, \tag{30}$$

where $\tau$ represents some threshold value and $\beta_i$ is an asymptotic lower bound on the probability that the regret of solution $\mathbf{x}_i$ is less than $\tau$. 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 $\mathbb{E}_{\mathbb{P}}[R(\tilde{\mathbf{z}}, \mathbf{x}_i)]$ for any given solution $\mathbf{x}_i$.

## 4. Numerical Experiments

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

1. Toy problem: comparison with related methods of Calafiore and Campi (2005) and Yanıkoğlu and den Hertog (2013), with additional analysis of ROBIST when the amount of data and the dimension of the problem increases.
2. Portfolio management problem: comparison with the robust optimization approach of Bertsimas et al. (2018).
3. Weighted distribution problem: comparison with the scenario optimization-based methods of Calafiore and Campi (2005), Carè et al. (2014), Calafiore (2016) and Garatti et al. (2022).
4. Two-stage adaptive lot-sizing problem: comparison with Vayanos et al. (2012).

For all experiments we utilize synthetic, randomly generated, data and split the data equally and randomly into the training 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 $\upsilon = 0.01$.

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 same toy problem as in Section 2.2, but now in $k$ dimensions. The problem is formulated as follows:

$$\max_{\mathbf{x}} \ \mathbf{e}^\intercal \mathbf{x} \tag{31}$$

$$\text{s.t. } \mathbf{z}^\intercal \mathbf{x} \leq 1, \tag{32}$$

$$\mathbf{x} \leq 1, \tag{33}$$

where $\mathbf{x}, \mathbf{z} \in \mathbb{R}^k$, and $\mathbf{e}^\intercal = (1, 1, \dots, 1) \in \mathbb{R}^k$. 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 Related Methods.** In the following experiments, we compare ROBIST with the methods proposed by Calafiore and Campi (2005), which is abbreviated as C&C, and Yanıkoğlu and den Hertog (2013), abbreviated as Y&dH. For the numerical experiments presented in this subsection, we set $\epsilon = 0.05$ and $\alpha = 0.01$.

For C&C we utilize Theorem 1 from Campi and Garatti (2008) to determine the number of randomly sampled scenarios with which (SCP) is solved. For Y&dH we use the modified $\chi^2$-distance as the $\phi$-divergence function, along with an ellipsoidal uncertainty set with an initial radius of 0.1 and a step size of 0.01. Furthermore, in our implementation of Y&dH we construct the cells such that the support of $\tilde{\mathbf{z}}$ given by $[-1,1]^k$ is divided into $10^k$ cells of equal geometry.

We first highlight a key difference between these three methods in the simple setting with $k = 2$. For this setting, C&C requires solving (SCP) with 130 randomly sampled scenarios. For ROBIST and Y&dH, we assume to have access to $N = 1{,}000$ randomly generated realizations of $\tilde{\mathbf{z}}$ and let both algorithms perform 105 iterations (the number of iterations needed for Y&dH to obtain a solution with feasibility certificate $\geq 0.95$). The resulting trade-off curves obtained from the three methods are depicted in Figure 4.

While the one-shot approach of C&C offers only a single solution, the iterative approaches of Y&dH and ROBIST offer 4 and 16 solutions of interest, respectively. The trade-off curve provided by ROBIST is richer than that of Y&dH due to two reasons. First, the method of Y&dH is restricted to the set of solutions attainable via solving the robust counterparts of uncertainty sets of a particular shape (in this case ellipsoidal). ROBIST generates solutions by optimizing over a finite set of scenarios, which allows for more variation in the resulting solutions. Second, in Y&dH the feasibility certificates are derived via the use of "cells" that approximate the support of $\tilde{\mathbf{z}}$. In contrast, ROBIST derives feasibility certificates by directly utilizing the testing data (without any intermediate approximation), which is more precise.

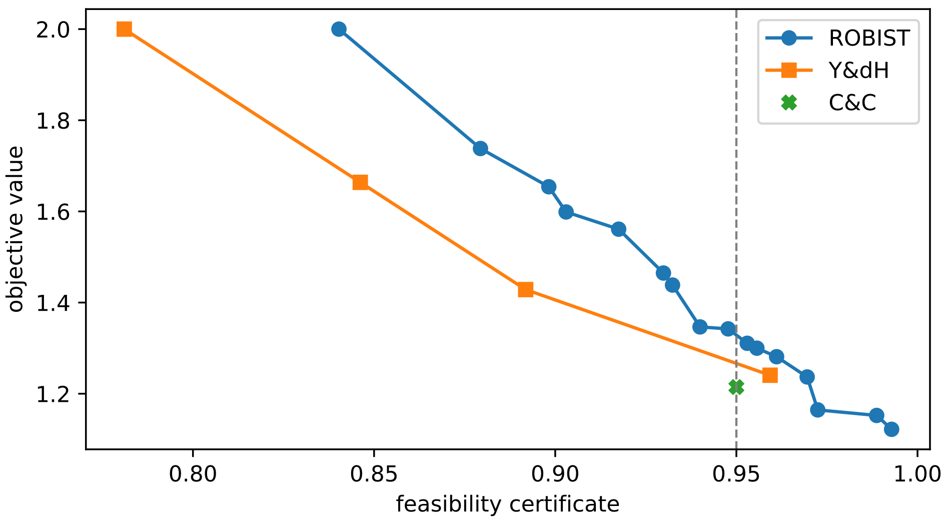


**Figure 4** **Trade-off curves constructed from the set of non-dominated solutions found by applying Calafiore and Campi (2005), Yanıkoğlu and den Hertog (2013) and ROBIST to Problem (31)-(33) with $k = 2$. The vertical dashed line represents the desired probability of constraint satisfaction $(1 - \epsilon)$.**

Two major limitations mentioned in Yanıkoğlu 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, where we apply the three methods to higher-dimensional problems. Here, $k$ represents the number of decision variables as well as the number of uncertain parameters. For C&C, the number of decision variables determines the required number of randomly sampled scenarios with which to solve (SCP). For Y&dH, it is the number of uncertain parameters that influences the number of "cells" and thus the amount of data required.[2] For both methods, these values are represented by $N$ in Table 1. For ROBIST there is no strict minimum or maximum regarding the amount of data and the algorithm is given access to $N = 1{,}000$ randomly generated realizations of $\tilde{\mathbf{z}}$.

In Table 1 we report the total computation time for each method, 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.

**Table 1 Results from applying Calafiore and Campi (2005), Yanıkoğlu and den Hertog (2013) and ROBIST to Problem** (31)-(33) **as the dimension of the problem (represented by** $k$**) increases. Here,** $N$ **indicates the amount of data utilized by the respective methods.**

| | $N$ | | | Computation time (s) | | | Objective value | | | Feasibility certificate | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| $k$ | C&C | Y&dH | ROBIST | C&C | Y&dH | ROBIST | C&C | Y&dH | ROBIST | C&C | Y&dH | ROBIST |
| 2 | 130 | 1,000 | 1,000 | 0.2 | 4 | 5 | 1.16 | 1.19 | 1.33 | 0.950 | 0.972 | 0.954 |
| 3 | 165 | 10,000 | 1,000 | 0.2 | 7 | 5 | 1.34 | 1.42 | 1.63 | 0.950 | 0.958 | 0.951 |
| 4 | 198 | 100,000 | 1,000 | 0.3 | 27 | 7 | 1.53 | 1.67 | 1.83 | 0.950 | 0.952 | 0.951 |
| 5 | 229 | 1,000,000 | 1,000 | 0.4 | 200 | 7 | 1.71 | 1.85 | 2.06 | 0.950 | 0.951 | 0.952 |

Even for the relatively small problem instances considered in Table 1, we find that the amount of data required by Y&dH quickly becomes unmanageable and the computational performance of the method deteriorates. For C&C the required number of randomly sampled scenarios is still manageable and the resulting (SCP) remains solvable within reasonable computation time (in Section 4.3, we consider larger problems for which this is no longer the case).

[2] We adhere to the rule of thumb stated in Yanıkoğlu 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 $5 \times 10^k$ 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).

We find that ROBIST is able to provide solutions with higher objective values than the solutions generated by C&C and Y&dH, while possessing comparable feasibility certificates. Even though the target probability of constraint satisfaction was set to $1-\epsilon = 0.95$, we find, by using additional out-of-sample testing (with $N = 10^6$), that the average empirical probability of constraint satisfaction for solutions generated by C&C and Y&dH is much higher at 0.981 and 0.984, respectively. While satisfying the same minimal probability guarantee, 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 the Optimality of the Solutions and the Accuracy of the Feasibility Certificates.** 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 (31)-(33), which allows us to analytically derive the true probability that (32) is satisfied:

$$p^*(\mathbf{x}) := \mathbb{P}\left(\tilde{\mathbf{z}}^\intercal \mathbf{x} \leq 1\right) = \frac{1}{2} + \frac{1}{2x_k}. \tag{34}$$

Furthermore, to expand the feasible region of the problem, we slightly alter constraint (33), which becomes $\mathbf{x} \leq k$. Therefore, given knowledge of the true distribution of $\tilde{\mathbf{z}}$, one could solve the following optimization problem:

$$\theta^* := \max_{\mathbf{x}} \left\{ \mathbf{e}^\intercal \mathbf{x} : \frac{1}{2} + \frac{1}{2x_k} \geq 1-\epsilon,\ \mathbf{x} \leq k,\ 1 + \sum_{j=1}^{k-1} x_j \leq x_k \right\}, \tag{35}$$

to obtain an optimal, sufficiently robust, solution. In the following sets of experiments we utilize (34) and (35) to assess the robustness and optimality of solutions obtained via Algorithm 1.

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

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

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

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

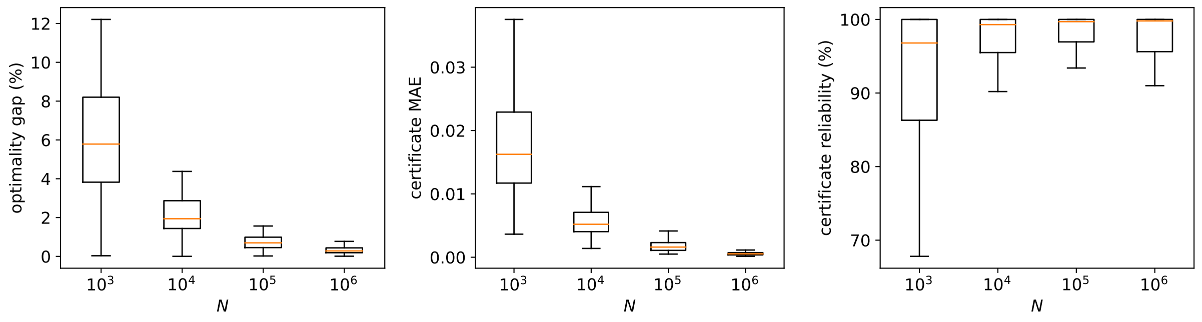
$$\frac{1}{i_{\max}} \sum_{i=1}^{i_{\max}} |p^*(\mathbf{x}_i) - \gamma_i|;$$

3. reliability of the feasibility certificates:

$$\frac{1}{i_{\max}}\sum_{i=1}^{i_{\max}} \mathbb{1}_{[\gamma_i \leq p^*(\mathbf{x}_i)]}.$$

The MAE gives an indication of the sharpness of our probability guarantees, while the reliability is the empirically observed frequency that the guarantees are truthful. The results for $N \in \{10^3, 10^4, 10^5, 10^6\}$ are presented in Figure 5.

**Figure 5 Box plots displaying three metrics regarding the performance of ROBIST when applied to a slightly altered version of Problem** (31)-(33) **with** $k = 2$**. We plot the results as a function of the amount of available data (**$N$**).**



(a) Optimality gap (%) (b) Mean absolute error (c) Reliability (%)

In Figures 5a and 5b we see a very similar trend as $N$ increases. While the two metrics might seem unrelated at first glance, the sharpness of the guarantees plays an important role in reducing conservativeness and thus closing the optimality gap. Figure 5c suggests that for $N = 10^3$, the reliability of the certificates can be quite dependent on the random data generation and subsequent random split into training and testing data sets. However, for $N \geq 10^4$, the average reliability is consistently above 90%. We note that there is no significant increase in computation time as $N$ increases as one is able to very efficiently compute $f(\mathbf{z}, \mathbf{x}_i), \forall \mathbf{z} \in \mathcal{D}^{\text{train}}_{N_1} \cup \mathcal{D}^{\text{test}}_{N_2}$ at each iteration $i$.

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

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

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

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

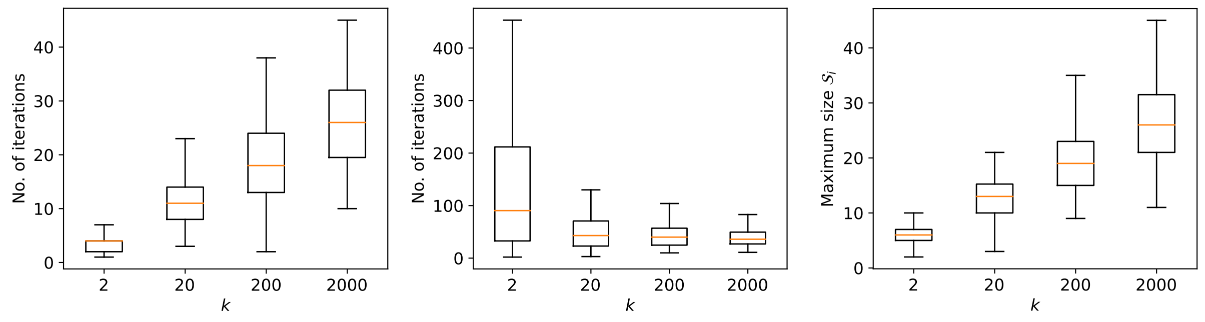
$$i_{\text{opt}} = \min\left\{i : \frac{\theta^* - g(\mathbf{x}_i)}{\theta^*} < 0.01,\ \gamma_i \geq 1 - \epsilon, p^*(\mathbf{x}_i) \geq 1 - \epsilon\right\};$$

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

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

We utilize the same setup as before ($\epsilon = 0.05$, $\alpha = 0.10$ and $i_{\max} = 1000$), but with a fixed number of data points ($N = 10^6$). The results of these experiments are displayed in Figure 6.

**Figure 6** **Box plots displaying three metrics regarding the performance of ROBIST when applied to a slightly altered version of Problem** (31)-(33) **of varying size (**$k$**).**



(a) $i_{\text{robust}}$ (b) $i_{\text{opt}}$ (c) $|\mathcal{S}_i|^{\max}$

The most striking finding is that $i_{\text{robust}}$ and $|\mathcal{S}_i|^{\max}$ both increase at similar, modest rates as the problem size ($k$) increases. Important to note is that this increase is not proportional to the increase in $k$: while $k$ increases 1000-fold, the maximum size of $|\mathcal{S}_i|$ becomes only 4.3 times as large (on average). In Figure 6b we observe that 100 iterations is, in many cases, sufficient for ROBIST to obtain near optimal solutions. Somewhat surprisingly, we also find that, on average, $i_{\text{opt}}$ decreases as $k$ increases. We suspect that this is due to the fact that the magnitude of the objective values increases as $k$ increases, as a result the number of near-optimal solutions also increases, which makes it relatively easier for ROBIST to find such a solution.

### 4.2. 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:

$$\max_{\mathbf{x}\geq\mathbf{0}} \ \mathbf{z}^\intercal\mathbf{x} \tag{36}$$

$$\text{s.t. } \mathbf{e}^\intercal\mathbf{x} = 1. \tag{37}$$

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 (36)-(37) using an epigraph reformulation, we obtain the following:

$$\max_{\mathbf{x},\theta} \; \theta \tag{38}$$

$$\text{s.t. } \mathbf{z}^\intercal \mathbf{x} \geq \theta, \tag{39}$$

$$\mathbf{e}^\intercal \mathbf{x} = 1, \tag{40}$$

$$\mathbf{x} \geq \mathbf{0}. \tag{41}$$

We note that solving (38)-(41) while providing a probability guarantee for Constraint (39) is equivalent to maximizing the value at risk (VaR), or quantile, of the portfolio, as:

$$\mathbb{P}(\tilde{\mathbf{z}}^\intercal \mathbf{x} \geq \theta) \geq 1-\epsilon \iff \text{VaR}_\epsilon^{\mathbb{P}}(\tilde{\mathbf{z}}^\intercal \mathbf{x}) \leq \theta. \tag{42}$$

**4.2.1. 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:

$$z_i = \begin{cases} \frac{\sqrt{(1-\gamma_i)\gamma_i}}{\gamma_i} & \text{with probability } \gamma_i \\ -\frac{\sqrt{(1-\gamma_i)\gamma_i}}{1-\gamma_i} & \text{with probability } 1-\gamma_i \end{cases}, \quad \gamma_i = \frac{1}{2}\left(1 + \frac{i}{k+1}\right), \quad i = 1, \ldots, k. \tag{43}$$

In this model, all assets $i = 1, \ldots, k$ have mean return 0%, standard deviation 1%, but have different skew and support. The higher indexed assets have a larger $\gamma_i$ 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 $i_{max} = 500$ 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 2 we report the average out-of-sample 10%-VaR over 100 repetitions.

**Table 2** **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 | 0.237 |
| 2000 | -1.095 | -0.411 | -0.396 | -0.451 | 0.243 |

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 7. Here we find that the solutions found by ROBIST put all wealth in either asset 9 or 10.

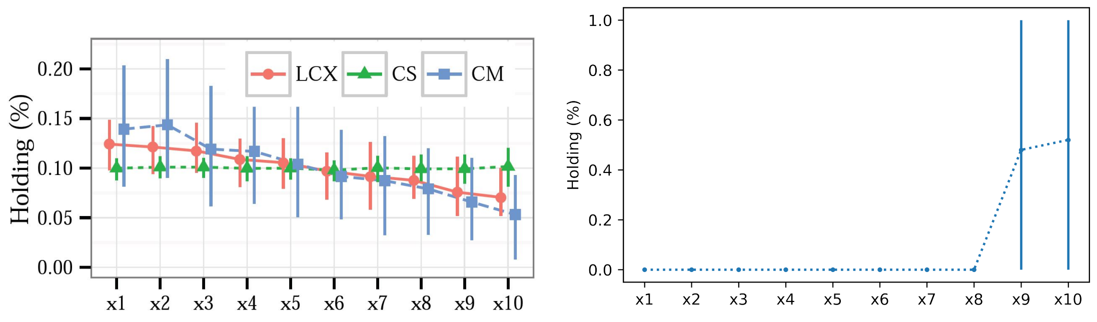


**Figure 7** **Graphical display of the average, along with 10% and 90% quantiles, of the portfolio holdings by method across the 100 repetitions of the experiment with $N = 500$. On the left we show the portfolio holdings reported in Figure 4 of Bertsimas et al. (2018), on the right we display the same 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:

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

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 the Value-at-Risk 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 the well-known danger in optimizing Value-at-Risk, namely that it does not account for the magnitude of losses that occur with probability less than $\epsilon$. Alternatively one may consider optimizing the conditional Value-at-Risk instead; see Basak and Shapiro (2001) and Laeven and Stadje (2014). We note that this is also possible with ROBIST (see Section 3.3).

### 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:

$$\max_{\mathbf{x}} \sum_{k=1}^{n} u_k \min\left\{\sum_{j=1}^{m} p_{jk}x_{jk}, d_k\right\} - \sum_{j=1}^{m}\sum_{k=1}^{n} c_{jk}x_{jk} - \sum_{k=1}^{n} h_k \max\left\{\sum_{j=1}^{m} p_{jk}x_{jk} - d_k, 0\right\} \tag{44}$$

$$\text{s.t.} \sum_{k=1}^{n} x_{jk} \leq a_j, \quad j = 1\dots, m, \tag{45}$$

$$x_{jk} \geq 0, \qquad j = 1\dots, m,\ k = 1\dots, n. \tag{46}$$

We note that this is a difficult problem to deal with using conventional robust optimization techniques, since (44) 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 Section 4.2, the uncertainty occurs only in the objective. Hence, ROBIST is slightly altered (see Section 3.1 for the details).

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 2016) 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 (2016) to determine $N_o^{RSD}$. For ISO we use Algorithm 2 of Garatti et al. (2022) to determine the set sizes $N_0^{ISO}, N_1^{ISO}, \ldots, N_{mn}^{ISO}$. 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 in Tables 3 and 4. To limit the effect of randomness, we report the average over 10 experiments, except in the cases where the computation time exceeds 1 hour (in these cases only a single experiment is performed). Furthermore, we restrict all methods to a maximum time limit of 10 hours.

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.

**Table 3 Comparison between Calafiore and Campi (2005), Carè et al. (2014), Calafiore (2016), 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 (44)-(46) 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.**

| | | $N$ | | | | | $\|\mathcal{S}\|^{max}$ | | | | | Computation time (s) | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| $m$ | $n$ | C&C | FAST | RSD | ISO | ROBIST | C&C | FAST | RSD | ISO | ROBIST | C&C | FAST | RSD | ISO | ROBIST |
| 5 | 10 | 10,580 | 3,062 | 28,662 | 5,678 | 3,000 | 10,580 | 1,000 | 6,017 | 5,678 | 61.2 | 708 | 11 | 245 | 17,367 | 46 |
| 10 | 20 | 34,918 | 6,073 | 41,733 | - | 3,000 | 34,918 | 4,000 | 26,160 | - | 92.9 | 27,541 | 314 | 12,438 | $>36{,}000$ | 114 |
| 15 | 30 | 74,468 | 11,073 | - | - | 3,000 | 74,468 | 9,000 | 60,586 | - | 117.0 | $>36{,}000$ | 3,382 | $>36{,}000$ | $>36{,}000$ | 211 |

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 4 Comparison between Calafiore and Campi (2005), Carè et al. (2014), Calafiore (2016), 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 (44)-(46) 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.**

| | | Objective value | | | | | Out-of-sample 1%-VaR | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| $m$ | $n$ | C&C | FAST | RSD | ISO | ROBIST | C&C | FAST | RSD | ISO | ROBIST |
| 5 | 10 | 458.8 | 446.0 | 463.3 | 458.7 | 468.3 | 475.3 | 475.5 | 474.8 | 471.7 | 476.2 |
| 10 | 20 | 973.2 | 949.2 | 974.6 | - | 979.1 | 1000.0 | 988.6 | 997.5 | - | 988.8 |
| 15 | 30 | - | 1465.8 | - | - | 1501.4 | - | 1505.9 | - | - | 1514.9 |

### 4.4. 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:

$$\begin{aligned}
&\min_{\mathbf{x},\theta} \theta && && (47)\\
&\text{s.t.} \sum_{i=1}^{m} c_i x_i + V(\mathbf{z},\mathbf{x}) \leq \theta, && && (48)\\
&0 \leq x_i \leq k_i, && i = 1.\ldots,m, && (49)
\end{aligned}$$

where:

$$\begin{aligned}
V(\mathbf{z},\mathbf{x}) := \min_{\mathbf{y}\geq \mathbf{0}} &\sum_{i=1}^{m}\sum_{j=1}^{m} t_{ij} y_{ij} && && (50)\\
\text{s.t. } & x_i + \sum_{j=1}^{n} y_{ji} - \sum_{j=1}^{n} y_{ij} \geq d_i, && i = 1,\ldots,m. && (51)
\end{aligned}$$

If Constraint (51) 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 (4), with $\gamma \geq 1-\epsilon$, for Constraint (48).

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 $t_{ij}$ is uncertain, the majority of existing RO methods can not be applied to this problem.

**4.4.1. 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 (47)-(51) to the following single-stage form:

$$\min_{\mathbf{x},A,\theta} \theta \tag{52}$$

$$\text{s.t.} \sum_{i=1}^{m} c_i x_i + \sum_{i=1}^{m}\sum_{j=1}^{m} t_{ij} y_{ij} \le \theta, \tag{53}$$

$$x_i + \sum_{j=1}^{n} y_{ji} - \sum_{j=1}^{n} y_{ij} \ge d_i, \qquad i = 1,\dots,m, \tag{54}$$

$$\mathbf{y} = A\mathbf{b}(\mathbf{z}), \tag{55}$$

$$\mathbf{y} \ge \mathbf{0}, \tag{56}$$

$$0 \le x_i \le k_i, \qquad i = 1.\dots,m. \tag{57}$$

Problem (52)-(57) can then be solved with respect to some set of randomly sampled scenarios (let this set be denoted as $\mathcal{S}_{VKR}$), where constraints (53)-(56) 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 $\epsilon$, $\alpha$ 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 (20)-(23), which involves $1 + m + m^2|\mathcal{S}_i|$ decision variables instead of the original $1 + m + m^2$ variables used in defining Problem (47)-(51). In these experiments we set $N = 1000$ and $i_{max} = 50$.

Setting $\epsilon = \alpha = 0.05$, we compare the two approaches to Problem (47)-(51) as the number of nodes ($m$) increases. The numerical results (average over 10 replications) are presented in Tables 5 and 6.

In Table 5 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 5 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 (47)-(51) with a varying number of nodes $m$.**

| | $\lvert\mathcal{S}_{VKR}\rvert$ | | ROBIST | | Computation time (s) | | |
|---|---|---|---|---|---|---|---|
| $m$ | $p = 1$ | $p = 2$ | $N$ | $\max_i \lvert\mathcal{S}_i\rvert$ | VKR$_{p=1}$ | VKR$_{p=2}$ | ROBIST |
| 2 | 809 | 2,655 | 1,000 | 10.2 | 5 | 60 | 366 |
| 3 | 2,783 | 26,103 | 1,000 | 13.5 | 56 | $>$ 3,600 | 407 |
| 4 | 10,853 | 117,162 | 1,000 | 13.9 | 2,214 | $>$ 3,600 | 425 |
| 5 | 24,774 | 392,584 | 1,000 | 15.5 | $>$ 3,600 | $>$ 3,600 | 439 |

In Table 6 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 6 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 (47)-(51) 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.**

| | Objective value | | | Out-of-sample 5%-VaR | | |
|---|---|---|---|---|---|---|
| $m$ | VKR$_{p=1}$ | VKR$_{p=2}$ | ROBIST | VKR$_{p=1}$ | VKR$_{p=2}$ | ROBIST |
| 2 | 795 | 792 | 729 | 788 | 778 | 728 |
| 3 | 1,192 | - | 1,032 | 1,178 | - | 1,013 |
| 4 | 1,588 | - | 1,222 | 1,555 | - | 1,198 |
| 5 | - | - | 1,367 | - | - | 1,322 |

## 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 $N \to \infty$) and are therefore only approximately valid. As such, ROBIST performs best when there is a large amount of data available.

## 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. This research was funded in part by the Netherlands Organization for Scientific Research under grant NWO VICI 2020-2027 (Laeven and Jin).

## A. $\phi$-divergence and confidence set

In order to formally evaluate the robustness of solutions, we use statistical testing that is based on *$\phi$-divergences.* Given two vectors $\mathbf{p}, \mathbf{q} \in \mathbb{R}^{d_\mathbf{p}}$, a $\phi$-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 $\phi : [0, \infty) \to \mathbb{R}$ is a convex function satisfying $\phi(1) = 0$, $\phi(a/0) := a \lim_{t\to\infty} \phi(t)/t$ for $a > 0$ and $\phi(0/0) = 0$. Using the modified $\chi^2$-distance, as we do throughout this paper, corresponds to choosing $\phi(t) = (t-1)^2$. An extensive study of the statistical properties of $\phi$-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 $N \to \infty$) to a chi-squared distribution with $d_\mathbf{p} - 1$ degrees of freedom. Here, $\phi''(1)$ denotes the second derivative of $\phi$ evaluated at 1. Hence, one can construct the following $(1-\alpha)$-confidence set for the true probability vector $\mathbf{p}^*$, as a $\phi$-divergence ball around the empirical estimate $\hat{\mathbf{p}}$:

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

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

## B. Proofs

### 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 $r = \frac{1}{N}\chi^2_{1,1-\alpha}$ 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 $\gamma_i$ 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 $\gamma_i$ is increasing in $p$.

## B.2. Proof of Lemma 2

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

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

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 $\{\breve{\mathcal{S}}_1,\ldots,\breve{\mathcal{S}}_M\}$:

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

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 $\breve{\mathbf{x}}^*$ 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 $\breve{\mathcal{S}}_j$, there is always a probability of removal and hence a path to the empty set $\breve{\mathcal{S}}_0$, which we denote as $\breve{\mathcal{S}}_j\to\breve{\mathcal{S}}_0$.

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

$$\breve{\mathcal{S}}_0\to\cdots\to\mathcal{S}^{\text{opt}}.$$

Indeed, let $\breve{\mathbf{x}}^*_0$ be the solution of the empty set $\breve{\mathcal{S}}_0$. Since $\min_{\mathbf{x}\in\mathcal{X}}\{g(\mathbf{x})\}\le\min_{\mathbf{x}\in\mathcal{X}}\{g(\mathbf{x}):f(\hat{\mathbf{z}}^j,\mathbf{x})\le 0,\forall\hat{\mathbf{z}}^j\in\breve{\mathcal{S}}_{k^*}\}$, we have that if $\breve{\mathbf{x}}^*_0$ is feasible for all scenarios in $\breve{\mathcal{S}}_{k^*}$, then $\breve{\mathbf{x}}^*_0$ must also be the unique optimal solution (uniqueness by assumption or by Remark 1) of solving (SCP) with $S_i=\breve{\mathcal{S}}_{k^*}$. In that case, we have by definition that $\breve{\mathbf{x}}^*_0$ is an optimal solution with respect to the test data and thus implies $\breve{\mathcal{S}}_0\in\mathcal{S}^{\text{opt}}$. Therefore, without loss of generality, we may assume that $\breve{\mathbf{x}}^*_0$ is infeasible for at least one scenario, say $\hat{\mathbf{z}}^q$ of $\breve{\mathcal{S}}_{k^*}$. Since there is a positive probability of adding this scenario, there is a positive probability path $\breve{\mathcal{S}}_0\to\breve{\mathcal{S}}_0\cup\{\hat{\mathbf{z}}^q\}$ and we can now repeat the same argument above for the solution $\breve{\mathbf{x}}^*$ of $\breve{\mathcal{S}}_0\cup\{\hat{\mathbf{z}}^q\}$: if $\breve{\mathbf{x}}^*$ is feasible for all scenarios in $\breve{\mathcal{S}}_{k^*}$, then $\breve{\mathcal{S}}_0\cup\{\hat{\mathbf{z}}^q\}\in\mathcal{S}^{\text{opt}}$. Otherwise, there is a positive probability of adding another scenario of $\breve{\mathcal{S}}_{k^*}$. This argument continues until either the subset $\breve{\mathcal{S}}_{k^*}$ 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).

## References

Alamo T, Tempo R, Luque A, Ramirez DR (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 DB (2010) A soft robust model for optimization under ambiguity. *Operations Research* 58(4-part-2):1220–1234.

Ben-Tal A, Brekelmans R, Den Hertog D, Vial JP (2017) Globalized robust optimization for nonlinear uncertain inequalities. *INFORMS Journal on Computing* 29(2):350–366.

Ben-Tal A, den Hertog D, de Waegenaere A, Melenberg B, Rennen G (2013) Robust solutions of optimization problems affected by uncertain probabilities. *Management Science* 59(2):341–357.

Ben-Tal A, El Ghaoui L, Nemirovski A (2009) *Robust Optimization* (Princeton university press).

Bertsimas D, Brown DB, Caramanis C (2011) Theory and applications of robust optimization. *SIAM Review* 53(3):464–501.

Bertsimas D, de Ruiter F (2016) Duality in two-stage adaptive linear optimization: Faster computation and stronger bounds. *INFORMS Journal on Computing* 28(3):500–511.

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(4):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(2):877–918.

Bertsimas D, Gupta V, Kallus N (2018) Data-driven robust optimization. *Mathematical Programming* 167(2):235–292.

Bertsimas D, Sim M (2004) The price of robustness. *Operations Research* 52(1):35–53.

Birge JR (1997) State-of-the-art-survey – Stochastic programming: Computation and applications. *INFORMS Journal on Computing* 9(2):111–133.

Birge JR, Louveaux F (2011) *Introduction to Stochastic Programming* (Springer Science & Business Media).

Boyd SP, Vandenberghe L (2004) *Convex Optimization* (Cambridge University Press).

Calafiore GC (2013) Direct data-driven portfolio optimization with guaranteed shortfall probability. *Automatica* 49(2):370–380.

Calafiore GC (2016) Repetitive scenario design. *IEEE Transactions on Automatic Control* 62(3):1125–1137.

Calafiore GC, Campi MC (2005) Uncertain convex programs: Randomized solutions and confidence levels. *Mathematical Programming* 102(1):25–46.

Calafiore GC, Ghaoui LE (2006) On distributionally robust chance-constrained linear programs. *Journal of Optimization Theory and Applications* 130:1–22.

Campi MC, Garatti S (2008) The exact feasibility of randomized solutions of uncertain convex programs. *SIAM Journal on Optimization* 19(3):1211–1230.

Campi MC, Garatti S (2018) Wait-and-judge scenario optimization. *Mathematical Programming* 167(1):155–189.

Carè A, Garatti S, Campi MC (2014) Fast—fast algorithm for the scenario technique. *Operations Research* 62(3):662–671.

Charnes A, Cooper W (1959) Chance-constrained programming. *Management Science* 6(1):73–79.

Delage E, Ye Y (2010) Distributionally robust optimization under moment uncertainty with application to data-driven problems. *Operations Research* 58(3):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 MC (2022) Complexity is an effective observable to tune early stopping in scenario optimization. *IEEE Transactions on Automatic Control* 68(2):928–942.

Guzman YA, Matthews LR, Floudas CA (2016) New a priori and a posteriori probabilistic bounds for robust counterpart optimization: I. unknown probability distributions. *Computers & Chemical Engineering* 84:568–598.

Hanasusanto GA, Roitch V, Kuhn D, Wiesemann W (2015) A distributionally robust perspective on uncertainty quantification and chance constrained programming. *Mathematical Programming* 151:35–62.

Jiang R, Guan Y (2016) Data-driven chance constrained stochastic program. *Mathematical Programming* 158(1-2):291–327.

Laeven RJ, Stadje M (2014) Robust portfolio choice and indifference valuation. *Mathematics of Operations Research* 39:1109–1141.

Mohajerin Esfahani P, Kuhn D (2018) Data-driven distributionally robust optimization using the wasserstein metric: Performance guarantees and tractable reformulations. *Mathematical Programming* 171(1):115–166.

Natarajan K, Pachamanova D, Sim M (2008) Incorporating asymmetric distributional information in robust value-at-risk optimization. *Management Science* 54(3):573–585.

Nemirovski A (2012) On safe tractable approximations of chance constraints. *European Journal of Operational Research* 219(3):707–718.

Nemirovski A, Juditsky A, Lan G, Shapiro A (2009) Robust stochastic approximation approach to stochastic programming. *SIAM Journal on Optimization* 19(4):1574–1609.

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(3):538–545.

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(3):814–833.

Postek K, den Hertog D, Melenberg B (2016) Computationally tractable counterparts of distributionally robust constraints on risk measures. *SIAM Review* 58(4):603–650.

Rahimian H, Mehrotra S (2019) Frameworks and results in distributionally robust optimization. *Open Journal of Mathematical Optimization* 3(4):1–85.

Roos E, den Hertog D (2020) Reducing conservatism in robust optimization. *INFORMS Journal on Computing* 32(4):1109–1127.

Shang C, You F (2020) A posteriori probabilistic bounds of convex scenario programs with validation tests. *IEEE Transactions on Automatic Control* 66(9):4015–4028.

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 RL (1984) Efficient monte carlo procedures for generating points uniformly distributed over bounded regions. *Operations Research* 32(6):1296–1308.

Vayanos P, Kuhn D, Rustem B (2012) A constraint sampling approach for multi-stage robust optimization. *Automatica* 48(3):459–471.

Wallace SW, Ziemba WT (2005) *Applications of Stochastic Programming* (SIAM).

Wang I, Becker C, Van Parys B, Stellato B (2022) Mean robust optimization. *arXiv preprint arXiv:2207.10820* .

Yanıkoğlu İ, den Hertog D (2013) Safe approximations of ambiguous chance constraints using historical data. *INFORMS Journal on Computing* 25(4):666–681.