POLITECNICO DI TORINO
Repository ISTITUZIONALE

Repetitive Scenario Design

Original
Repetitive Scenario Design / Calafiore, Giuseppe Carlo. - In: IEEE TRANSACTIONS ON AUTOMATIC CONTROL. - ISSN 0018-9286. - STAMPA. - 62:3(2017), pp. 1125-1137. [10.1109/TAC.2016.2575859]

Availability:
This version is available at: 11583/2645176 since: 2017-03-22T11:22:12Z

Publisher:
IEEE

Published
DOI:10.1109/TAC.2016.2575859

Terms of use:

This article is made available under terms and conditions as specified in the corresponding bibliographic description in the repository

Publisher copyright
IEEE postprint/Author's Accepted Manuscript

©2017 IEEE. Personal use of this material is permitted. Permission from IEEE must be obtained for all other uses, in any current or future media, including reprinting/republishing this material for advertising or promotional purposes, creating new collecting works, for resale or lists, or reuse of any copyrighted component of this work in other works.

(Article begins on next page)

12 October 2024

Repetitive Scenario Design

Giuseppe C. Calafiore, *Senior Member, IEEE*

***Abstract*—Repetitive Scenario Design (RSD) is a randomized approach to robust design based on iterating two phases: a standard scenario design phase that uses N scenarios (design samples), followed by randomized feasibility phase that uses N_o test samples on the scenario solution. We give a full and exact probabilistic characterization of the number of iterations required by the RSD approach for returning a solution, as a function of N, N_o, and of the desired levels of probabilistic robustness in the solution. This novel approach broadens the applicability of the scenario technology, since the user is now presented with a clear tradeoff between the number N of design samples and the ensuing expected number of repetitions required by the RSD algorithm. The plain (one-shot) scenario design becomes just one of the possibilities, sitting at one extreme of the tradeoff curve, in which one insists in finding a solution in a single repetition: this comes at the cost of possibly high N. Other possibilities along the tradeoff curve use lower N values, but possibly require more than one repetition.**

***Keywords*—Scenario design, probabilistic robustness, randomized algorithms, random convex programs.**

I. Introduction

The purpose of the approach described in this paper is to obtain a *probabilistically reliable* solution for some design problem affected by uncertainty. The concept of "probabilistic design" has been discussed extensively in the control community in the last two decades, and it is now well accepted as a standard tool for tackling difficult robust design problems; we refer the reader to the survey paper [5] and to the book [18] for many pointers to the related literature. The essential elements of a probabilistic design approach are the following ones:

1) A *spec function*, $f(\theta, q) : \mathbb{R}^n \times \mathbb{Q} \to \mathbb{R}$, which associates a real value to each pair (θ, q) of a design parameter $\theta \in \mathbb{R}^n$ and uncertainty instance $q \in \mathbb{Q}$, where $\mathbb{Q} \subseteq \mathbb{R}^{n_q}$. Function f represents the design constraints and specifications of the problem and, in particular, we shall say that a design θ is a *robust* design, if $f(\theta, q) \leq 0$, $\forall q \in \mathbb{Q}$. In this paper, we make the standing assumption that f is convex in θ, while arbitrary dependence in q is allowed.
2) A probability measure Prob defined on $\mathbb{Q}$, which describes the probability distribution of the uncertainty.

Equipped with these two essential elements, for given $\epsilon \in (0, 1)$, and given design vector θ, we are in position to define the *probability of violation* for the spec function at θ:

$$V(\theta) \doteq \text{Prob}\{q \in \mathbb{Q} : \ f(\theta, q) > 0\}. \tag{1}$$

Giuseppe C. Calafiore, Dipartimento di Automatica e Informatica, Politecnico di Torino, Italy. giuseppe.calafiore@polito.it

We say that θ is an *ϵ-probabilistic robust design*, if it holds that $V(\theta) \leq \epsilon$. Further, a designer also typically seeks to minimize some cost function of θ (which can be considered of the linear form $c^\top \theta$, without loss of generality; see, e.g., Section 8.3.4.4 in [6]), while guaranteeing that $V(\theta) \leq \epsilon$. Finding such an ϵ-probabilistic robust design amounts to solving a so-called *chance-constrained* optimization problem, which is computationally hard in general, and perhaps harder than finding a classical deterministic robust design. Chance-constrained optimization problems can be solved exactly only in very restrictive cases (e.g., when f is linear, and q has some specific distribution, such as Normal; see, e.g., [16]). Deterministic convex *approximations* of chance-constrained problems are discussed in [13] for some special classes of problems where f is affine in q and the entries of q are independent. Also, the *sampling average approximation* (SAA) method replaces the probability constraint $V(\theta) \leq \epsilon$ with one involving the empirical probability of violation based on N sampled values of q; see, e.g., [12], [14]. The optimization problem resulting from SAA, however, is non-convex and intractable, in general, even when the original function f is convex in θ, as it is assumed in the present work.

A. The standard scenario theory

While effective approximation schemes for chance-constrained optimization problems remain to date hard to tackle numerically, an alternative and efficient randomized scheme emerged in the last decade for finding ϵ-probabilistic robust designs. This technique, which is now a well-established technology (see, e.g., the recent surveys [9], [15]) in the area of robust control, is called "scenario design," and was introduced in [3]. In scenario design one considers N i.i.d. random samples of the uncertainty $\{q^{(1)}, \ldots, q^{(N)}\} \doteq \omega$, and builds a scenario *random convex program* (RCP):

$$\begin{aligned} \min_{\theta \in \Theta} \quad & c^\top \theta \\ \text{s.t.:} \quad & f(\theta, q^{(i)}) \leq 0, \quad i = 1, \ldots, N, \end{aligned} \tag{2}$$

where Θ is some given convex and compact domain, and c is the given objective direction. An optimal solution θ^* to this problem, if it exists, is a random variable which depends on the multiextraction ω, i.e., $\theta^* = \theta^*(\omega)$. As a consequence, the violation probability relative to a scenario solution, $V(\theta^*)$, is itself, a priori, a random variable.

Scenario design lies somewhere in between worst-case robust design (where $c^\top \theta$ is minimized subject to $f(\theta, q) \leq 0$ for all $q \in \mathbb{Q}$) and chance-constrained design (where $c^\top \theta$ is minimized subject to $V(\theta) \leq \epsilon$). Indeed, the optimal objective value resulting from a scenario design is lower than the worst-case optimal objective, and it is (with high probability) higher than the optimal objective a related chance-constrained

problem (see, e.g., Section 6 in [2]). Moreover, a fundamental feature of scenario design is that its optimal solution $\theta^*(\omega)$ is feasible with high probability for the chance-constrained problem. This key result is recalled next for the sake of clarity. We shall work under the following simplifying assumption, which is routinely made in the literature on scenario design; see [3], [7].

Assumption 1: With probability (w.p.) one with respect to the multi-extraction $\omega = \{q^{(1)}, \ldots, q^{(N)}\}$, problem (2) is feasible and it attains a unique optimal solution $\theta^*(\omega)$. $\star$

Also, we need the following standard definition (see, e.g., Definition 4 in [3]).

Definition 1: Let $J^* = c^\top \theta^*$ denote the optimal objective value of problem (2). Also, for $j = 1, \ldots, N$, define

$$\begin{aligned} J_j^* \doteq \min_{\theta \in \Theta} \quad & c^\top \theta \\ \text{s.t.:} \quad & f(\theta, q^{(i)}) \leq 0, \quad i \in \{1, \ldots, N\} \setminus j. \end{aligned}$$

The j-th constraint in (2) is said to be a *support constraint* if $J_j^* < J^*$. $\star$

A key fact is that, regardless of the problem structure and of N, the number of support constraints for problem (2) cannot exceed n (the number of decision variables); see, e.g., Theorem 3 in [3]. If an instance of problem (2) happens to have precisely n support constraints, then the problem instance is said to be *fully supported (f.s.)*; see Definition 3 in [7], and Definition 2.5 in [2]. If the instances of problem (2) are fully supported almost surely with respect to the random extraction ω of the N constraints, then we say that problem (2) is fully supported w.p. one. The following key result holds, see Theorem 1 in [7], and Corollary 3.4 in [2].

Theorem 1: Let Assumption 1 hold. Then, for given $\epsilon \in [0, 1]$ and $N \geq n$, it holds that

$$\begin{aligned} F_V(\epsilon) &\doteq \text{Prob}^N\{\omega : V(\theta^*(\omega)) \leq \epsilon\} && (3) \\ &\geq \sum_{i=n}^{N} \binom{N}{i} \epsilon^i (1-\epsilon)^{N-i} && (4) \\ &\doteq 1 - \beta_\epsilon(N). \end{aligned}$$

Moreover, the bound (4) is tight, since it holds with equality for the class of problems of the form (2) that are fully supported with probability one. $\star$

A remarkable feature of the result in (4) is that it holds irrespective of the probability distribution assumed on q, and that it depends on the problem structure only through the dimension parameter n. Notice that the quantity $\beta_\epsilon(N)$ represents a Binomial cumulative distribution, as formally defined later in Eq. (6).

B. Scenario problems and Bernoulli trials

For given $\epsilon \in [0, 1]$ and $N \geq n$, let us consider the following Bernoulli variable associated to problem (2):

$$z = z(\omega) = \begin{cases} 1, & \text{if } V(\theta^*(\omega)) \leq \epsilon \\ 0, & \text{otherwise.} \end{cases}$$

By the definition in Eq. (3), the event $z = 1$ happens w.p. $F_V(\epsilon)$. One interpretation of Eq. (4) is thus that each time we solve a scenario problem (2) we have an a priori probability $\geq 1 - \beta_\epsilon(N)$ of realizing a "successful design," that is of finding a solution θ^* which is an ϵ-probabilistic robust design, and a probability $\leq \beta_\epsilon(N)$ of realizing a "failure," that is of finding a solution θ^* which is not ϵ-probabilistic robust.

In the classical scenario theory it is usually prescribed to choose N so to make $\beta_\epsilon(N)$ very small (values as low as 10^{-12} are common). This guarantees that the event $\{V(\theta^*(\omega)) \leq \epsilon\}$ will happen with "practical certainty." In other words, in such a regime, the scenario problem will return an ϵ-probabilistic robust solution with practical certainty. Moreover, a key feature of scenario theory is that such high level of confidence can be reached at a relatively "cheap" computational price. Indeed, considering the condition $\beta_\epsilon(N) \leq \beta$ for some given desired probability level $\beta \in (0, 1)$, and using some fairly standard techniques for bounding the Binomial tail (see, e.g., Corollary 5.1 in [2] for the details), one can prove that the condition is satisfied for[1]

$$N \geq \frac{2}{\epsilon} \left(\ln \beta^{-1} + n - 1 \right). \tag{5}$$

Since β^{-1} appears in the above bound under a logarithm, we indeed see that N grows gracefully with the required certainty level β^{-1}. However, there are cases in which the number N of constraints prescribed by (5) for reaching the desired confidence levels is just too high for practical numerical solution. Convex optimization solvers are certainly efficient, but there are practical limits on the number of constraints they can deal with; these limits depend on the actual type of convex problem (say, a linear program (LP), or a semidefinite program (SDP)) one deals with. A critical situation is, for instance, when problem (2) is a semidefinite program (formally, f can be taken as the maximum eigenvalue function of the matrices describing the linear inequality constraints), since dealing with SDP problems with many thousands of LMI constraints can pose serious practical issues.

C. Contribution

In this paper we discuss how a variation of the scenario approach can be used for obtaining an ϵ-probabilistic robust solution with high confidence, using "small" values of N. More precisely, we are interested in using scenario optimization in a regime of N for which the right-hand side of Eq. (4) is *not* close to one. We shall do so by solving repeatedly instances of the scenario problem, and checking the result via a suitable "violation oracle." This novel approach, named *repetitive scenario design* (RSD), is discussed in Section II, which contains all the relevant results. Section III describes two numerical examples of robust control design where the proposed approach is applied. For improving readability, technical proofs are reported in the Appendix.

[1]Notice that the expression in (5) may be conservative; the exact minimal value of N can be easily found numerically by searching for the least integer N such that $\sum_{i=n}^{N} \binom{N}{i} \epsilon^i (1-\epsilon)^{N-i} \geq 1 - \beta$.

D. Notation and preliminaries

We shall make intensive use of the beta and related probability distributions. Some definitions and standard facts are recalled next. We denote by $\text{beta}(\alpha,\beta)$ the beta density function with parameters $\alpha > 0$, $\beta > 0$:

$$\text{beta}(\alpha,\beta;t) \doteq \frac{1}{B(\alpha,\beta)} t^{\alpha-1}(1-t)^{\beta-1}, \quad t \in [0,1],$$

where $B(\alpha,\beta) \doteq \frac{\Gamma(\alpha)\Gamma(\beta)}{\Gamma(\alpha+\beta)}$, and Γ is the Gamma function (for α,β integers, it holds that $B(\alpha,\beta)^{-1} = \alpha\binom{\alpha+\beta-1}{\beta-1}$). Also, we denote by $\text{Fbeta}(\alpha,\beta)$ the cumulative distribution function of the $\text{beta}(\alpha,\beta)$ density:

$$\text{Fbeta}(\alpha,\beta;t) \quad \doteq \quad \int_0^t \text{beta}(\alpha,\beta;\vartheta)\mathrm{d}\vartheta, \quad t \in [0,1].$$

$\text{Fbeta}(\alpha,\beta;t)$ is the regularized incomplete beta function, and a standard result establishes that, for α,β integers, it holds that

$$\text{Fbeta}(\alpha,\beta;t) = \sum_{i=\alpha}^{\alpha+\beta-1} \binom{\alpha+\beta-1}{i} t^i (1-t)^{\alpha+\beta-1-i}.$$

The number x of successes in d independent Bernoulli trials, each having success probability p, is a random variable with Binomial distribution (which we denote by $\text{Bin}(d,p)$); its cumulative distribution is given by

$$\begin{aligned}
\text{Prob}\{x \le z\} &= \text{Prob}\{x \le \lfloor z \rfloor\} = \sum_{i=0}^{\lfloor z \rfloor} \binom{d}{i} t^i (1-t)^{d-i} \\
&= 1 - \sum_{i=\lfloor z \rfloor+1}^{d} \binom{d}{i} t^i (1-t)^{d-i} \\
&= 1 - \text{Fbeta}(\lfloor z \rfloor + 1, d - \lfloor z \rfloor; t) \\
&\le 1 - \text{Fbeta}(z+1, d-z; t) \qquad (6) \\
&= \text{Fbeta}(d-z, z+1; 1-t),
\end{aligned}$$

where $\lfloor z \rfloor$ denotes the largest integer no larger than z. The number x of successes in d binary trials, where each trial has success probability p, and p is itself a random variable with $\text{beta}(\alpha,\beta)$ distribution, is a random variable with a so-called beta-Binomial distribution f_{bb}: for $i = 0,1,\ldots,d$,

$$f_{\text{bb}}(d,\alpha,\beta;i) \doteq \binom{d}{i} \frac{B(i+\alpha, d-i+\beta)}{B(\alpha,\beta)}. \qquad (7)$$

The cumulative distribution of a beta-Binomial random variable is given by (see, e.g., [11], [19])

$$\begin{aligned}
F_{\text{bb}}(d,\alpha,\beta;z) &\doteq \text{Prob}\{x \le z\} = \sum_{i=0}^{\lfloor z \rfloor} f_{\text{bb}}(d,\alpha,\beta;i) \\
&= 1 - \frac{1}{d+1} \frac{B(\beta+d-z-1, \alpha+z+1)}{B(\alpha,\beta)B(d-z,z+2)} F_d(a,b;z),
\end{aligned}$$

where $F_d(a,b;z)$ is the generalized hypergeometric function

$${}_3F_2(1, \alpha+z+1, -n+z+1; z+2, -\beta-n+z+2; 1).$$

II. Repetitive scenario design

This section develops the main idea of this paper. By *repetitive scenario design* (RSD) we here mean an iterative computational approach in which, at each iteration k, the scenario problem (2) is solved and then the ensuing solution θ_k^* is checked by a violation oracle (either deterministic, or randomized, as illustrated next). If the oracle returns `false`, another iteration is performed; if instead the oracle returns `true`, the algorithm is terminated and the current solution θ_k^* is returned.

In the RSD the user selects a desired probabilistic feasibility level $\epsilon \in (0,1)$, and a number $N \ge n$ of scenarios to be used in (2). We have from Theorem 1 that, at any iteration k, it holds that

$$\text{Prob}^N\{\omega^{(k)} : V(\theta_k^*) \le \epsilon\} = F_V(\epsilon) \ge 1 - \beta_\epsilon(N), \qquad (8)$$

where $\omega^{(k)}$ denotes the multisample $\{q_k^{(1)}, \ldots, q_k^{(N)}\}$. In very elementary terms, each iteration of the RSD method can be thought of as a biased "coin toss," where the probability of a success in a toss (that is, of getting θ_k^* such that $V(\theta_k^*) \le \epsilon$) is at least $1 - \beta_\epsilon(N)$. In our setting, this probability need not be too close to one: the simple idea behind the RSD method is to repeat the coin toss until we obtain a success, where success is detected by the violation oracle. As one may easily argue intuitively, the probability of obtaining a success at some point in the algorithm is much higher than the probability of obtaining a success in a single toss. A similar idea has been recently proposed in [8], where the authors solve repeatedly a "reduced-size" scenario problem, followed by a randomized test of feasibility. The approach and the results in [8], however, are distinctively different from the ones proposed here. In [8], the scenario problems are solved using a number N_k of scenarios that grows with the iteration count k, up to the value N_{plain} that corresponds to the plain, one-shot, scenario design. The major shortcoming of the approach in [8] is that no theoretical analysis is offered for the number of iterations required by their algorithm, and no tradeoff curve is proposed for the choice of N_k in function of the expected running time of the algorithm. As a result, there is no a-priori deterministic or probabilistic guarantee that the algorithm does not reach the final iteration, in which N_k equals N_{plain}, hence the worst-case complexity of the algorithm in [8] can be worse than the one of the plain scenario design method, and an actual reduction of the number of design samples is not theoretically guaranteed by the approach in [8].

We shall next analyze the probabilistic features of our RSD algorithm in two cases. In the first case we assume that an ideal exact feasibility oracle is available for checking the current solution θ_k^*; this case may be unrealistic in general, but serves for providing an insightful preliminary analysis of the RSD approach. In the second case, we analyze the RSD approach when a practically implementable randomized feasibility oracle is used.

A. Violation oracles

A *deterministic ϵ-violation oracle* (ϵ-DVO) is a "black box" which, when given in input a value of the design variable θ,

returns as output a flag value which is `true` if $V(\theta) \leq \epsilon$, and `false` otherwise. Such an oracle may *not* be realizable computationally in practice, since computing the probability in (1) is numerically hard, in general. For this reason, we next also introduce a *randomized ϵ'-violation oracle* (ϵ'-RVO), which is defined by means of the randomized scheme described next.

ϵ'-RVO (*Randomized ϵ'-violation oracle*) Input data: integer N_o, level $\epsilon' \in [0,1]$, and $\theta \in \mathbb{R}^n$. Output data: a logic flag, `true` or `false`.

1) Generate N_o i.i.d. samples $\omega_o \doteq \{q^{(1)}, \ldots, q^{(N_o)}\}$, according to Prob.
2) For $i = 1, \ldots, N_o$, let $v_i = 1$ if $f(\theta, q^{(i)}) > 0$ and $v_i = 0$ otherwise.
3) If $\sum_i v_i \leq \epsilon' N_o$, return `true`, else return `false`.

The ϵ'-RVO simply evaluates the empirical probability of violation on N_o test samples, and returns `true` if it is below ϵ', and `false` otherwise. A similar type of randomized feasibility oracle has been previously introduced in [4], and used in a probabilistic design setting also in [5]; see also Section 11.1 in [18], and the "validation" step proposed in [8]. However, the ϵ'-RVO we propose in this paper is different from the one used in the cited references: the latter exits with a `false` flag as soon as one infeasible sample is found, whereas the ϵ'-RVO allows up to $\lfloor \epsilon' N_o \rfloor$ infeasible samples before exit. Also, the kind of a priori analysis we develop here for the repetitive scenario design based on the ϵ'-RVO is entirely novel.

B. Repetitive scenario design with ideal oracle

We consider the following RSD algorithm, in which each repetition consists of a plain scenario optimization step, followed by a feasibility check of the ensuing solution, performed by an exact feasibility oracle.

Algorithm 1 (RSD with ϵ-DVO): Input data: integer $N \geq n$, level $\epsilon \in [0,1]$. Output data: solution θ^*. Initialization: set iteration counter to $k = 1$.

1) (Scenario step) Generate N i.i.d. samples $\omega^{(k)} \doteq \{q_k^{(1)}, \ldots, q_k^{(N)}\}$ according to Prob, and solve scenario problem (2). Let θ_k^* be the resulting optimal solution.
2) (ϵ-DVO step) If $V(\theta_k^*) \leq \epsilon$, then set flag to `true`, else set it to `false`.
3) (Exit condition) If flag is `true`, then exit and return current solution $\theta^* \leftarrow \theta_k^*$; else set $k \leftarrow k+1$ and goto 1.

$\star$

The following theorem holds.

Theorem 2: Let Assumption 1 hold. Given $\epsilon \in [0,1]$ and $N \geq n$, define the running time K of Algorithm 1 as the value of the iteration counter k when the algorithm exits. Then:

1) The solution θ^* returned by Algorithm 1 is an ϵ-probabilistic robust design, i.e., $V(\theta^*) \leq \epsilon$.
2) The expected running time of Algorithm 1 is $\leq (1 - \beta_\epsilon(N))^{-1}$, and equality holds if the scenario problem is f.s. w.p. 1.
3) The running time of Algorithm 1 is $\leq k$ with probability $\geq 1 - \beta_\epsilon(N)^k$, and equality holds if the scenario problem is f.s. w.p. 1.

$\star$

See Section A in the Appendix for a proof of Theorem 2.

Remark 1 (Potential and limits of the RSD approach): The preliminary results in Theorem 2 show the potential of the RSD approach. Suppose that N is chosen so that $\beta_\epsilon(N)$ is, say, 0.4. This means that a plain (i.e., one-shot) scenario approach has only at least a 0.6 chance of returning a "good" solution (that is, a 0.6 probability of returning an ϵ-probabilistic robust design, i.e., a θ^* such that $V(\theta^*) \leq \epsilon$). However, we see from point 3 of Theorem 2 that there is, for instance, more than $1 - 10^{-9}$ probability that Algorithm 1 returns an ϵ-probabilistic robust design within 23 iterations. Further, the eventual outcome of Algorithm 1 is ϵ-probabilistic robust with probability one, and the expected number of iterations of the RSD algorithm is just $(1 - 0.4)^{-1} = 1.67$, in the worst case of a f.s. problem.

Theorem 2 also shows a fundamental limit of the RSD approach: we can decrease N (and hence increase $\beta_\epsilon(N)$) with respect to a plain scenario design approach, but we cannot decrease N too much, for otherwise $\beta_\epsilon(N) \to 1$, and the expected number of iterations of Algorithm 1 tends to ∞. There is thus a fundamental tradeoff between the reduction of N (which reduces the effort needed for solving the scenario problem) and the increase of the number of iterations of Algorithm 1. This tradeoff can be fully captured by plotting the expected running time bound $(1 - \beta_\epsilon(N))^{-1}$ versus the number N of scenarios.

C. Repetitive scenario design with randomized oracle

This section contains the main contribution of this paper. Here, we consider a realistically implementable version of the RSD approach, in which a randomized oracle is used instead of the ideal deterministic one.

Algorithm 2 (RSD with ϵ'-RVO): Input data: integers N, N_o, level $\epsilon' \in [0,1]$. Output data: solution θ^*. Initialization: set iteration counter to $k = 1$.

1) (Scenario step) Generate N i.i.d. samples $\omega^{(k)} \doteq \{q_k^{(1)}, \ldots, q_k^{(N)}\}$ according to Prob, and solve scenario problem (2). Let θ_k^* be the resulting optimal solution.
2) (ϵ'-RVO step) Call the ϵ'-RVO with current θ_k^* as input, and set flag to `true` or `false` according to the output of the ϵ'-RVO.
3) (Exit condition) If flag is `true`, then exit and return current solution $\theta^* \leftarrow \theta_k^*$; else set $k \leftarrow k+1$ and goto 1.

$\star$

A generic iteration, or stage, k, of Algorithm 2 is illustrated in Figure 1.

We next analyze Algorithm 2 along two directions. First, we observe that, contrary to Algorithm 1, the present Algorithm 2 may exit with a solution which is not ϵ-probabilistic robust. This is due to the randomized nature of the oracle, which may detect a "false positive," by misclassifying as "good" a

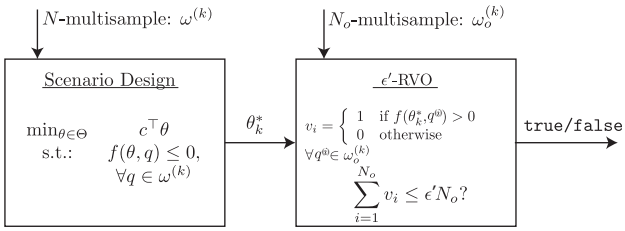

Fig. 1. Generic stage k of Algorithm 2.

solution θ_k^* for which instead $V(\theta_k^*)>\epsilon$. We show that the probability of such a "bad exit" event can be made arbitrarily small. Second, we fully characterize the probabilistic running time (iterations to exit) of the algorithm. We start with the following key preliminary lemma, which is the backbone of the whole paper.

Lemma 1: Let Assumption 1 hold and, for any given iteration k of Algorithm 2, define the events

$$\begin{aligned}\text{True} &= \{\epsilon'\text{-RVO returns } \texttt{true}\}\\ \text{GoodTrue} &= \{\epsilon'\text{-RVO returns } \texttt{true}\cap V(\theta_k^*)\le\epsilon\}\\ \text{BadTrue} &= \{\epsilon'\text{-RVO returns } \texttt{true}\cap V(\theta_k^*)>\epsilon\}\end{aligned}$$

Let

$$\begin{aligned}&f_{\rm bb}(N_o,n,N+1-n;i)\\ &\quad\doteq\binom{N_o}{i}\frac{B(i+n,N_o-i+N-n+1)}{B(n,N+1-n)},\\ &H_{\epsilon,\epsilon'}(N,N_o)\\ &\quad\doteq 1-\sum_{i=0}^{\lfloor\epsilon' N_o\rfloor} f_{\rm bb}(N_o,n,N+1-n;i)\cdot\\ &\qquad\cdot\,\text{Fbeta}(n+i,N+N_o-n-i+1;\epsilon),\\ &H_{1,\epsilon'}(N,N_o)\\ &\quad\doteq 1-\sum_{i=0}^{\lfloor\epsilon' N_o\rfloor} f_{\rm bb}(N_o,n,N+1-n;i),\\ &\bar\beta_{\epsilon,\epsilon'}(N,N_o)\\ &\quad\doteq\text{Fbeta}(N+(1-\epsilon')N_o-n+1,n+\epsilon' N_o;1-\epsilon).\end{aligned}$$

At any iteration k of Algorithm 2, it holds that

$$\text{Prob}^{N+N_o}\{\text{True}\}\ge 1-H_{1,\epsilon'}(N,N_o) \tag{9}$$

$$\text{Prob}^{N+N_o}\{\text{GoodTrue}\}\ge 1-H_{\epsilon,\epsilon'}(N,N_o) \tag{10}$$

$$\ge(1-\bar\beta_{\epsilon,\epsilon'}(N,N_o))(1-H_{1,\epsilon'}(N,N_o)). \tag{11}$$

$$\text{Prob}^{N+N_o}\{\text{BadTrue}\} \tag{12}$$

$$\le\text{Fbeta}((1-\epsilon')N_o,\epsilon' N_o+1;1-\epsilon)\beta_\epsilon(N).$$

Moreover, if problem (2) is f.s. w.p. one, then bounds (9) and (10) hold with equality, and

$$\begin{aligned}\text{Prob}^{N+N_o}\{\text{BadTrue}\} &= H_{\epsilon,\epsilon'}(N,N_o)-H_{1,\epsilon'}(N,N_o)\\ &\le\bar\beta_{\epsilon,\epsilon'}(N,N_o)(1-H_{1,\epsilon'}(N,N_o)).\end{aligned} \tag{13}$$

⋆

See Section B in the Appendix for a proof of Lemma 1.

We can now state the main result concerning Algorithm 2.

Theorem 3: Let Assumption 1 hold. Let $\epsilon,\epsilon'\in[0,1]$, $\epsilon'\le\epsilon$, and $N\ge n$ be given. Let all the notation be set as in Lemma 1, and let $\text{Prob}^{\times\times}$ denote the product probability $\text{Prob}^{N+N_o}\times\text{Prob}^{N+N_o}\times\cdots$. Define the event BadExit in which Algorithm 2 exits returning a "bad" solution θ^*:

$$\text{BadExit}\doteq\{\text{Algorithm 2 returns } \theta^*\text{: } V(\theta^*)>\epsilon\}.$$

The following statements hold.

1)

$$\begin{aligned}&\text{Prob}^{\times\times}\{\text{BadExit}\}\\ &\quad\le\frac{\text{Fbeta}((1-\epsilon')N_o,\epsilon' N_o+1;1-\epsilon)}{1-H_{1,\epsilon'}(N,N_o)}\beta_\epsilon(N).\end{aligned} \tag{14}$$

If problem (2) is f.s. w.p. one, then it actually holds that

$$\text{Prob}^{\times\times}\{\text{BadExit}\}\le\bar\beta_{\epsilon,\epsilon'}(N,N_o)). \tag{15}$$

2) The expected running time of Algorithm 2 is $\le(1-H_{1,\epsilon'}(N,N_o))^{-1}$, and equality holds if the scenario problem is f.s. w.p. 1.
3) The running time of Algorithm 2 is $\le k$ with probability $\ge 1-H_{1,\epsilon'}(N,N_o)^k$, and equality holds if the scenario problem is f.s. w.p. 1.

⋆

See Section C in the Appendix for a proof of Theorem 3.

1) Asymptotic bounds: A key quantity related to the expected running time of Algorithm 2 is $H_{1,\epsilon'}(N,N_o)$, which is the upper tail of a beta-Binomial distribution. This quantity is related to the hypergeometric function ${}_3F_2$, and to ratios of Gamma functions, which may be delicate to evaluate numerically for large values of the arguments. It is therefore useful to have a more "manageable," albeit approximate, expression for $H_{1,\epsilon'}(N,N_o)$. The following corollary gives an asymptotic expression for $H_{1,\epsilon'}(N,N_o)$, see Section D in the Appendix for a proof.

Corollary 1: For $N_o\to\infty$ it holds that

$$H_{1,\epsilon'}(N,N_o)\to\beta_{\epsilon'}(N). \tag{16}$$

⋆

An interesting consequence of Corollary 1 is that, for large N_o, and $\epsilon'\le\epsilon$, we have $H_{1,\epsilon'}(N,N_o)\simeq\beta_{\epsilon'}(N)\ge\beta_\epsilon(N)$, from which we conclude that

$$\hat K\doteq\frac{1}{1-H_{1,\epsilon'}(N,N_o)}\simeq\frac{1}{1-\beta_{\epsilon'}(N)}\ge\frac{1}{1-\beta_\epsilon(N)}. \tag{17}$$

This last equation gives us an approximate, asymptotic, expression for the upper bound $\hat K$ on the expected running time of Algorithm 2, and also tells us that, for $\epsilon'\le\epsilon$, this bound cannot be better (smaller) than the corresponding bound of the "ideal" Algorithm 1.

D. Practical dimensioning of the scenario and oracle blocks

In a typical probabilistic design problem we are given the dimension n of the decision variable and the level $\epsilon \in (0,1)$ of probabilistic robustness we require from our design. If we intend to use a randomized approach, we also set a confidence level $1-\beta \in (0,1)$, which is the a-priori level of probability with which our randomized approach will be successfull in returning an ϵ-probabilistic robust design. In a plain (i.e., non repetitive) scenario design setting, this requires dimensioning the number N of scenarios so to guarantee that $\beta_\epsilon(N) \leq \beta$; this can be done, for instance, by using the bound in (5), or via a simple numerical search over N. However, if the required N turns out to be too large in practice (e.g., the ensuing scenario optimization problem becomes impractical to deal with numerically), we can switch to a repetitive scenario design approach. In such a case, we suggest the following route for designing the scenario and oracle blocks. Let us first select a level $\epsilon' \leq \epsilon$ to be used in the oracle. Qualitatively, decreasing ϵ' increases the expected running time K and decreases the required N_o, and the converse happens for increasing ϵ'. We may suggest setting ϵ' in the range $[0.5, 0.9]\epsilon$.

1) Dimensioning the scenario block: Guidelines for the choice of N cannot be given in general terms, since the actual choice of a suitable N will depend on the specific type of optimization problem one deals with, and also on the software/hardware environment available for solving it. For instance, if the scenario problem is a linear program (LP), then larger values of N may be admissible, whereas if the scenario problem is an SDP then they user may not want to exceed with the value of N. Once this additional problem-specific and environment-specific information is available, we can dimension the scenario optimization block by choosing N so as to achieve a good tradeoff between the specific complexity of the scenario program (which grows with N) and the expected number of iterations required by the RSD approach (which decreases with N). This choice can be made, for instance, by plotting the approximate expression (which becomes exact as $N_o \to \infty$) in (17) for the upper bound on the expected running time of Algorithm 2, $(1-\beta_{\epsilon'}(N))^{-1}$, as a function of N, and selecting a value of N that achieves the desired tradeoff.

2) Dimensioning the oracle block: Once N has been selected according to the approach described above, we consider point 1 and point 2 in Theorem 3 and we dimension the ϵ'-RVO block by searching numerically for an N_o such that the right-hand side of (14) (or of (15), if the problem is f.s.) is $\leq \beta$.

Remark 2: We observe that, in general, the bound in (14) should be used for the design of the ϵ'-RVO block. However, the expression in (15) is easier to deal with than the one in (14). It is hence advisable to use the former in a preliminary dimensioning phase; the so-obtained values can then be verified ex-post against the actual bound in (14). Another advantage of (15) is that, using a bounding technique analogous to the one described in Section 5 of [2], we can "invert" the condition $\bar{\beta}_{\epsilon,\epsilon'}(N, N_o) \leq \beta$, finding (after some manipulation) that this condition is satisfied if

$$N_o\delta + N(\delta/2+\epsilon') \geq \frac{\epsilon}{\delta}\ln\beta^{-1} + n - 1, \quad \delta \doteq \epsilon - \epsilon' > 0. \quad (18)$$

With a choice of the pair (N, N_o) such that (18) is satisfied, we guarantee a priori that our randomized Algorithm 2 may fail in returning an ϵ-probabilistic robust design w.p. at most β, as desired (rigorously, this only holds under the assumption that the scenario problem is f.s. w.p. one). The nice feature highlighted by (18) is that now the "workload" necessary to achieve the desired failure level β is subdivided between N (samples in the scenario problem) and N_o (samples in the oracle): a lower complexity scenario problem can be employed, as long as it is paired with a randomized oracle having a suitable N_o. Notice, however, that, in making the choice of the (N, N_o) pair, the expected running time of Algorithm 2 should also be taken into account, and that this places a lower limit on how small N can be, see also the discussion in Section II-C1.

Remark 3: We further observe that, in typical cases, dealing with large N_o is a milder problem than dealing with large N. This is due to the fact that merely *checking* satisfaction of inequality $f(\theta_k^*, q^{(i)}) \leq 0$, for $i = 1, \ldots, N_o$, is generally easier than solving a related optimization problem with as many constraints. Also, we remark that the ϵ'-RVO algorithm is inherently parallel, so an M-fold speedup can potentially be gained if M processors are available in parallel for the randomized feasibility test. Actually, the whole approach can be formulated in a fully parallel – instead of sequential – form, where W workers solve in parallel W instances of scenario problems, and each worker has its own M parallel sub-workers to be used in the randomized oracle. Such a parallel version of the RSD method can be easily analyzed using the probabilistic tools developed in this paper.

III. Numerical examples

We exemplify the steps of the RSD approach, from algorithm dimensioning to numerical results, using two examples of robust control design. The first example deals with robust finite-horizon input design for an uncertain linear system, while the second example deals with robust performance design for a positive linear system.

A. Robust finite-horizon input design

We consider a system of the form

$$x(t+1) = A(q)x(t) + Bu(t), \quad t = 0, 1, \ldots; \; x(0) = 0,$$

where $u(t)$ is a scalar input signal, and $A(q) \in \mathbb{R}^{n_a, n_a}$ is an interval uncertain matrix of the form

$$A(q) = A_0 + \sum_{i,j=1}^{n_a} q_{ij} e_i e_j^\top, \quad |q_{ij}| \leq \rho, \; \rho > 0,$$

where e_i is a vector of all zeros, except for a one in the i-th entry. Given a final time $T \geq 1$ and a target state $\bar{x}$, the problem is to determine an input sequence $\{u(0), \ldots, u(T-1)\}$ such that *(i)* the state $x(T)$ is robustly contained in a small ball around the target state $\bar{x}$, and *(ii)* the input energy

$$A_0 = \begin{bmatrix} -0.7214 & -0.0578 & 0.2757 & 0.7255 & 0.2171 & 0.3901 \\ 0.5704 & 0.1762 & 0.3684 & -0.0971 & 0.6822 & -0.5604 \\ -1.3983 & -0.1795 & 0.1511 & 1.0531 & -0.1601 & 0.9031 \\ -0.6308 & -0.0058 & 0.4422 & 0.8169 & 0.512 & 0.2105 \\ 0.7539 & 0.1423 & 0.2039 & -0.3757 & 0.5088 & -0.6081 \\ -1.3571 & -0.1769 & 0.1076 & 1.0032 & -0.1781 & 0.9151 \end{bmatrix}, \quad B = \begin{bmatrix} 0 \\ 1 \\ 0 \\ 1 \\ 0 \\ 1 \end{bmatrix}$$

$\sum_k u(k)^2$ is not too large. We write $x(T) = x(T;q) = \mathcal{R}(q)u$, where $\mathcal{R}(q)$ is the T-reachability matrix of the system (for a given q), and $u \doteq (u(0), \ldots, u(T-1))$. Then, we formally express our design goals in the form of minimization of a level γ such that

$$\|x(T;q) - \bar{x}\|_2^2 + \lambda \sum_{t=0}^{T-1} u(t)^2 \leq \gamma,$$

where $\lambda \geq 0$ is a tradeoff parameter. Letting $\theta = (u, \gamma)$, the problem is formally stated in our framework by setting

$$f(\theta, q) \leq 0, \quad \text{where } f(\theta, q) \doteq \|\mathcal{R}(q)u - \bar{x}\|_2^2 + \lambda \|u\|_2^2 - \gamma.$$

Assuming that the uncertain parameter q is random and uniformly distributed in the hypercube $\mathbb{Q} = [-\rho, \rho]^{n_a \times n_a}$, our scenario design problem takes the form

$$\begin{aligned} \min_{\theta=(u,\gamma)} \quad & \gamma \\ \text{s.t.:} \quad & f(\theta, q^{(i)}) \leq 0, \quad i = 1, \ldots, N. \end{aligned}$$

a) Dimensioning the RSD algorithm: We set $T = 10$, thus the size of the decision variable $\theta = (u, \gamma)$ of the scenario problem is $n = 11$. We set the desired level of probabilistic robustness to $1 - \epsilon = 0.995$, i.e., $\epsilon = 0.005$, and require a level of failure of the randomized method below $\beta = 10^{-12}$, that is, we require the randomized method to return a good solution with "practical certainty." Using a plain (one-shot) scenario approach, imposing $\beta_\epsilon(N) \leq \beta$ would require $N \geq 10440$ scenarios. Let us now see how we can reduce this N figure by resorting to a repetitive scenario design approach.

Let us fix $\epsilon' = 0.7\epsilon = 0.0035$, thus $\delta = \epsilon - \epsilon' = 0.0015$. A plot of the (asymptotic) bound on expected number of iterations, $(1 - \beta_{\epsilon'}(N))^{-1}$, as a function of N is shown in Figure 2. A value of, say, $N = 2000$ is workable for the specific optimization problem at hand, and we see from the plot in Figure 2 that such choice would correspond to a value of about 10 for the upper bound on the expected number of iterations in Algorithm 2. Let then us choose the value of $N = 2000$ for the scenario block.

For $\beta = 10^{-12}$, the simplified condition in (18) tells us that $N_o \geq 62403$. Let us choose $N_o = 63000$ samples to be used in the oracle. With the above choices we have $H_{1,\epsilon'}(N, N_o) = 0.8963$, thus the algorithm's upper bound on average running time is $\hat{K} = (1 - H_{1,\epsilon'}(N, N_o))^{-1} = 9.64$. Notice that this upper bound is tight for f.s. problems, but it is conservative for problems that are not necessarily f.s. Thus, in general, we may expect a performance that is in practice better than the one predicted by the theoretical worst-case bound.

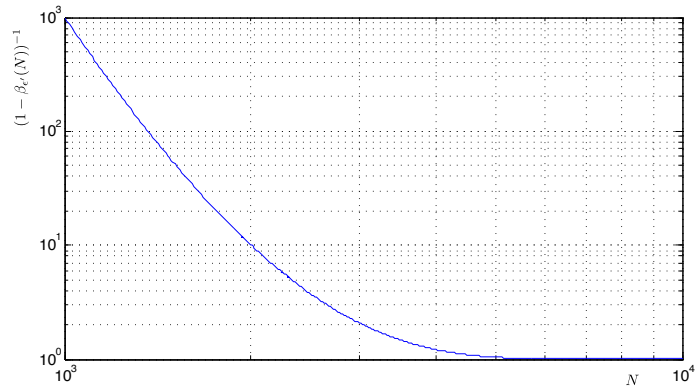

Fig. 2. Example in Section III-A: Log-log plot of $(1 - \beta_{\epsilon'}(N))^{-1}$ vs. N.

b) Numerical test: We considered the nominal matrix A_0 of dimension $n_a = 6$ and B matrix shown on top of this page, with target state $\bar{x} = [1, -1/2, 2, 1, -1, 2]^\top$, $\rho = 0.001$, and $\lambda = 0.005$. We run Algorithm 2 for 100 times, and on each test run we recorded the number of iterations and the solution returned upon exit. Figure 3(a) shows the number of repetitions in the test runs: we see that the algorithm exited most of the times in a single repetition, with a maximum of 4 repetitions, which is below the figure predicted by the upper bound $\hat{K} = 9.64$: practical performance was thus better than predicted, which suggests that the problem at hand is not fully supported w.p. 1. Figure 3(b) shows the level of empirical violation probability evaluated by the oracle upon exit. Finally, Figure 4(a) shows the optimal γ level returned by the algorithm in the test runs, and Figure 4(b) shown the optimal input signal returned by the algorithm, averaged over the 100 test runs.

c) Computational improvements: In this example, the RSD approach permitted a substantial reduction of the number of design samples (from the 10440 samples required by the plain scenario method, to just 2000 samples), at the price of a very moderate number of repetitions (the average number of repetitions in the 100 test runs was 1.27).

The numerical experiments were carried out on an Intel Xeon X5650 machine using CVX under Matlab; [10]. On average over the 100 test experiments, the RSD method (with $N = 2000$, $N_o = 63000$) required 224 s to return a solution. For comparison purposes, we also run a plain, one-shot, scenario optimization with the $N = 10440$ scenarios that are required to attain the desired $\beta = 10^{-12}$ level: the time required for obtaining such a solution was 2790 s. Using the RSD approach instead of a plain one-shot scenario design thus

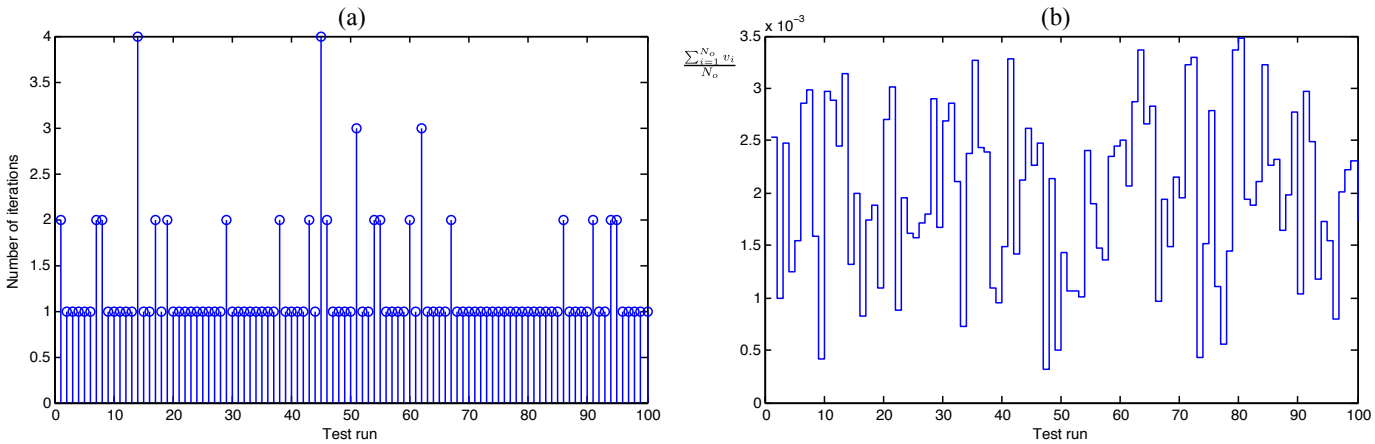

Fig. 3. Example in Section III-A: (a) Repetitions of Algorithm 2 in the 100 test runs. (b) Levels of empirical violation probability evaluated by the oracle upon exit, in the 100 test runs.

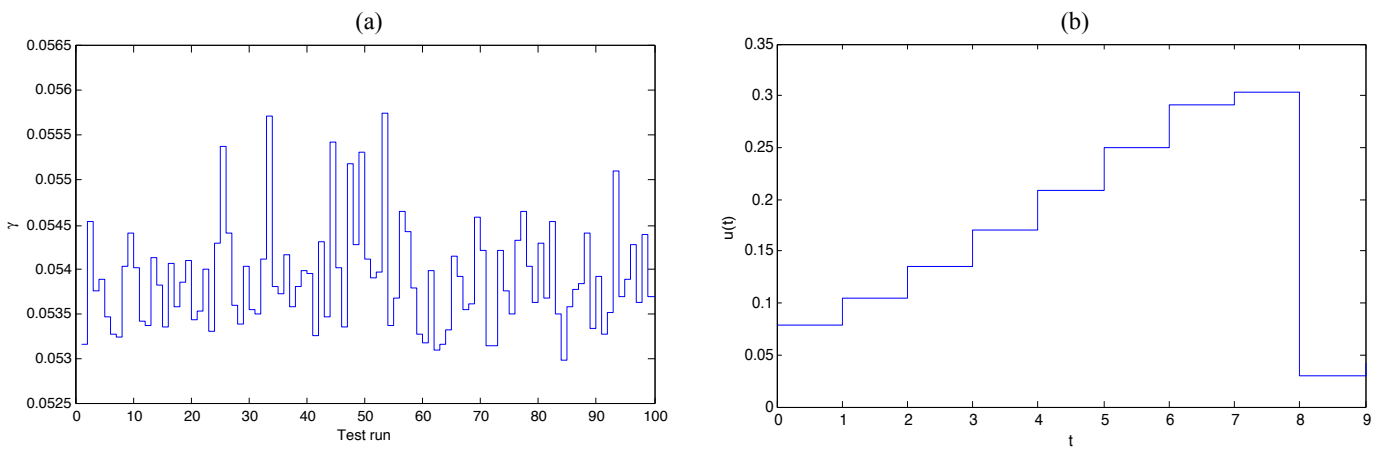

Fig. 4. Example in Section III-A: (a) Optimal γ level returned by Algorithm 2 in the 100 test runs. (b) Average over the 100 test runs of the optimal input $u(t)$ returned by Algorithm 2.

yielded a reduction in computing time of about one order of magnitude. The reason for this improvement is due to the fact that the scenario optimization problem in the RSD approach (which uses $N = 2000$ scenarios) took about 173 s to be solved on a typical run, and the subsequent randomized oracle test (with $N_o = 63000$) is computationally cheap, taking only about 3.16 s.

B. An uncertain linear transportation network

As a second example, we consider a variation on a transportation network model introduced in Section 3 of [17]; see Figure 5.

The model is described by state equations where the states x_i, $i = 1, \ldots, 4$, represent the contents of four buffers, the parameters $\ell_{ij} \geq 0$ represent the rate of transfer from buffer j to buffer i, $w(t) \geq 0$ is an input flow on the second buffer,

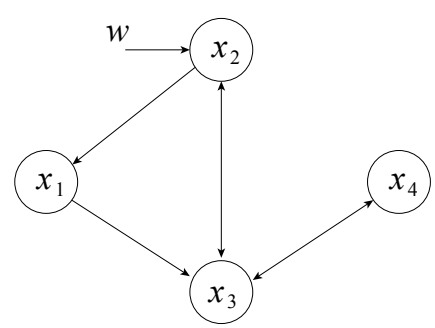

Fig. 5. Example in Section III-B: A network model.

and we take as output y the total content of the buffers; see Eq. (19)-(20).

$$\dot{x} = \begin{bmatrix} -1-\ell_{31} & \ell_{12} & 0 & 0 \\ 0 & -\ell_{12}-\ell_{32} & \ell_{23} & 0 \\ \ell_{31} & \ell_{32} & -\ell_{23}-\ell_{43} & \ell_{34} \\ 0 & 0 & \ell_{43} & -4-\ell_{34} \end{bmatrix} x + \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \end{bmatrix} w \tag{19}$$

$$y = \begin{bmatrix} 1 & 1 & 1 & 1 \end{bmatrix} x. \tag{20}$$

We consider the situation in which $\ell_{31} = 2 + q_1$, $\ell_{34} = 1 + q_2$, $\ell_{43} = 2 + q_3$, where $\ell = [\ell_{12}\ \ell_{23}\ \ell_{32}]^\top \in [0,1]^3$ is a vector of parameters to be designed, and $q = [q_1\ q_2\ q_3]^\top$ is an uncertainty term, which is assumed to be a truncated Normal random vector with zero mean, covariance matrix $\Sigma = 0.2^2 I$, and $\|q\|_\infty \leq 1$. This system has the form $\dot{x} = A(\ell, q)x + Bw$, $y = Cx$, where $B \geq 0$, $C \geq 0$ (element-wise), and the $A(\ell, q)$ matrix is Metzler (i.e., the off-diagonal entries of A are nonnegative). Theorem 4 in [17] states that, for given ℓ, q, this system is stable and the peak-to-peak gain from w to y is smaller that some given γ if and only if there exist $\xi \geq 0$ such that

$$\begin{bmatrix} A(\ell,q) & B \\ C & 0 \end{bmatrix} \begin{bmatrix} \xi \\ \mathbf{1} \end{bmatrix} < \begin{bmatrix} 0 \\ \gamma \mathbf{1} \end{bmatrix},$$

where $\mathbf{1}$ is a vector of ones. By taking N i.d.d. samples $q^{(i)}$ of q, a scenario design problem is one in which one seeks to minimize the peak-to-peak gain γ subject to the above constraint on the scenarios; see Eq. (21). This problem is a "robustified" version of the one discussed in Section V of [17]. The problem as stated is not convex, due to the product terms between entries in ξ and ℓ. However, by introducing new variables $\mu_{12} = \ell_{12}\xi_2$, $\mu_{32} = \ell_{32}\xi_2$, $\mu_{23} = \ell_{23}\xi_3$, we rewrite the problem as an LP in the variables ξ, $\mu = [\mu_{12}\ \mu_{32}\ \mu_{23}]^\top$, and γ; see Eq. (22).

d) Dimensioning the RSD algorithm: The size of the decision variable $\theta = (\xi, \mu, \gamma)$ of the scenario problem is $n = 8$. As in the previous example, we set the desired level of probabilistic robustness to $1 - \epsilon = 0.995$, i.e., $\epsilon = 0.005$, and require a level of failure of the randomized method below $\beta = 10^{-12}$. Using a plain (one-shot) scenario approach, imposing $\beta_\epsilon(N) \leq \beta$ would require $N \geq 9197$ scenarios. We next reduce this N figure by resorting to a repetitive scenario design approach.

Let us fix $\epsilon' = 0.7\epsilon = 0.0035$, thus $\delta = \epsilon - \epsilon' = 0.0015$. Plotting the asymptotic bound on expected number of iterations, $(1-\beta_{\epsilon'}(N))^{-1}$ as a function of N (as we did in Figure 2 for the previous example), we see that the choice $N = 1340$ corresponds to a value of about 10 for the upper bound on the expected number of iterations in Algorithm 2. Let us choose this value of N for the scenario block.

For $\beta = 10^{-12}$, the simplified condition in (18) tells us that $N_o \geq 62273$ samples can be used in the randomized feasibility oracle. With the above choices we have $H_{1,\epsilon'}(N, N_o) = 0.8931$, thus the algorithm's upper bound on average running time is $\hat{K} = (1 - H_{1,\epsilon'}(N, N_o))^{-1} = 9.36$ (notice again that, in general, we may expect a performance which is in practice better than the one predicted by this theoretical worst-case bound, since the the actual problem may not be fully supported).

e) Numerical test and computational performance: We first solved the problem via a plain scenario approach, using $N = 9197$ scenarios. The computational time was of about 50 s, resulting in the following optimal solution:

$$\xi = \begin{bmatrix} 0.2314 \\ 0.5000 \\ 1.7206 \\ 0.9763 \end{bmatrix}, \quad \mu = \begin{bmatrix} 0.5000 \\ 0.5000 \\ 0.0000 \end{bmatrix}, \quad \gamma = 3.4283.$$

Next, we run the RSD method (Algorithm 2, with $N = 1340$, $N_o = 62273$) for 100 times, and on each test run we recorded the number of iterations and the solution returned upon exit. Figure 6(a) shows the number of repetitions in the test runs: we see that the algorithm exited most of the times in a single repetition, with a maximum of 3 repetitions; average 1.24 repetitions. Figure 6(b) shows the level of empirical violation probability evaluated by the oracle upon exit. Finally, Figure 7 shows the optimal γ level returned by the algorithm in the test runs.

The average (over the 100 test trials) running time of the RSD method was about 6.4 s. Since the plain scenario approach required about 50 s, it was about 680% slower than the newly proposed RSD approach, in this test example. Each repetition of the RSD method required about 4.6 s for solving the scenario problem (with $N = 1340$), and 0.6 s for the randomized oracle check (with $N_o = 62273$); once again, we observe that the oracle time was much lower than the scenario optimization time.

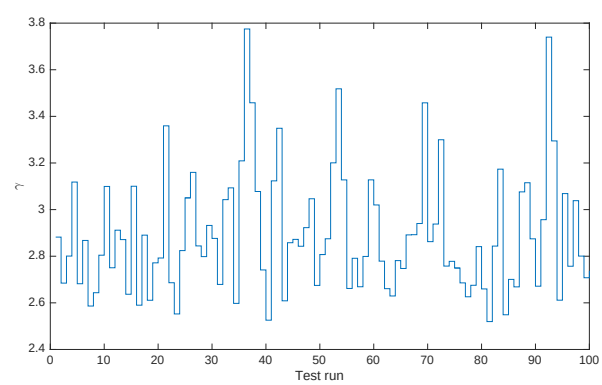

Fig. 7. Example in Section III-B: Optimal γ level returned by Algorithm 2 in the 100 test runs.

IV. Conclusions

Repetitive scenario design generalizes the scenario approach to robust design by setting up an iterative procedure whereby

$$\min_{\ell_{12},\ell_{23},\ell_{32}\in[0,1];\gamma\ \xi\geq 0} \quad \gamma \tag{21}$$
$$\text{s.t.:} \quad \begin{bmatrix} -3-q_1^{(i)} & \ell_{12} & 0 & 0 \\ 0 & -\ell_{12}-\ell_{32} & \ell_{23} & 0 \\ 2+q_1 & \ell_{32} & -\ell_{23}-2-q_3^{(i)} & 1+q_2^{(i)} \\ 0 & 0 & 2+q_3^{(i)} & -5-q_2^{(i)} \end{bmatrix} \xi + B\mathbf{1} < 0, \quad i=1,\ldots,N$$
$$C\xi < \gamma\mathbf{1}.$$

$$\min_{\mu\geq 0;\xi\geq 0;\gamma} \quad \gamma \tag{22}$$
$$\text{s.t.:} \quad \begin{bmatrix} -3-q_1^{(i)} & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 2+q_1^{(i)} & 0 & -2-q_3^{(i)} & 1+q_2^{(i)} \\ 0 & 0 & 2+q_3^{(i)} & -5-q_2^{(i)} \end{bmatrix} \xi + \begin{bmatrix} 1 & 0 & 0 \\ -1 & -1 & 1 \\ 0 & 1 & -1 \\ 0 & 0 & 0 \end{bmatrix} \mu + B\mathbf{1} < 0, \quad i=1,\ldots,N$$
$$C\xi < \gamma\mathbf{1}, \quad \mu_{12}\leq\xi_2,\ \mu_{32}\leq\xi_2,\ \mu_{23}\leq\xi_3.$$

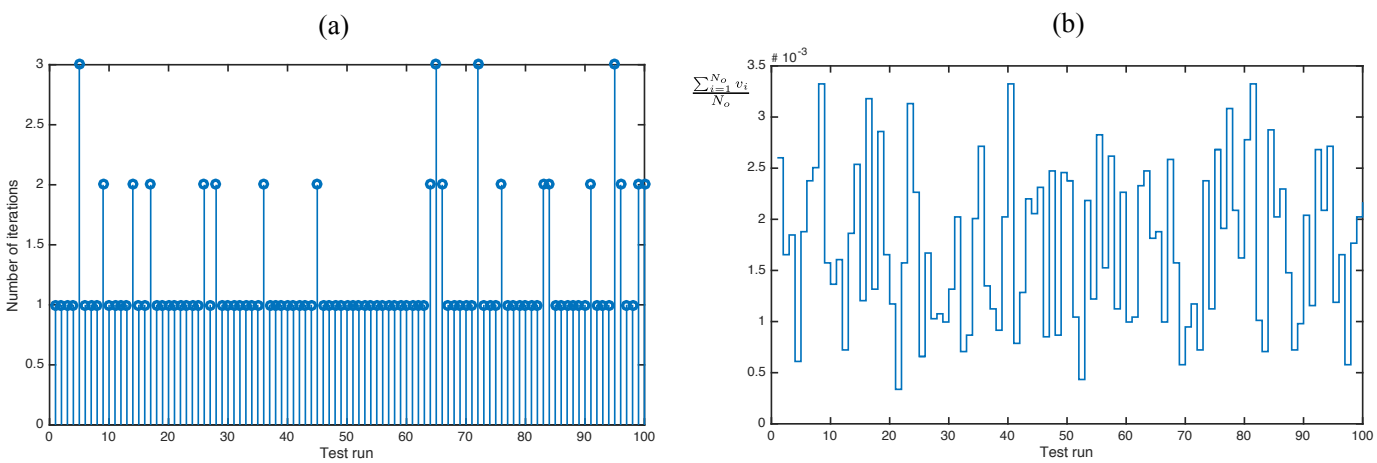

Fig. 6. Example in Section III-B: (a) Repetitions of Algorithm 2 in the 100 test runs. (b) Levels of empirical violation probability evaluated by the oracle upon exit, in the 100 test runs.

scenario design *trials* are followed by a randomized check on the feasibility level of the solution. The expected number of repetitions (or trials) in this procedure is dictated by the key quantity $H_{1,\epsilon'}(N,N_o)$, which is well approximated, for large N_o, by $\beta_{\epsilon'}(N)$. For $H_{1,\epsilon'}(N,N_o)\to 0$ we recover the extreme situation of the standard, one-shot, scenario design, in which a valid solution is found in a single repetition, at the cost of possibly large N. For smaller N values, we can trade off complexity in the solution of the scenario problem for additional iterations in the RSD algorithm. The extent to which N can be reduced is however limited by the upper bound $\hat{K}$ we impose on the expected running time, since Eq. (17) tells us that $H_{1,\epsilon'}(N,N_o)\simeq\beta_{\epsilon'}(N)\leq 1-\hat{K}^{-1}$. Numerical examples showed that the proposed RSD approach may indeed lead to sensible improvements in computational time, compared to a plain scenario approach.

Appendix

A. *Proof of Theorem 2*

The first point of the theorem is obvious, sice the algorithm terminates if and only if `true` is returned by the deterministic oracle, which happens if and only if the condition $V(\theta_k^*)\leq\epsilon$ is satisfied.

For point two, let $z_k = z_k(\omega^{(k)})$, $k=1,\ldots$, be i.i.d. Bernoulli variables representing the outcome of the ϵ-DVO step at each iteration, i.e., $z_k=1$ if $V(\theta_k^*)\leq\epsilon$ (oracle returns `true`), and $z_k=0$ otherwise (oracle returns `false`). From Eq. (8) we observe that the probability of $z_k=1$ is $F_V(\epsilon)\geq 1-\beta_\epsilon(N)$. Since the algorithm terminates as soon as a `true` is returned by the oracle, the *running*

time of the algorithm is defined as the random variable

$$K \doteq \{\text{iteration } k \text{ at which } \texttt{true} \text{ is returned for the first time}\}.$$

Clearly, K has a geometric distribution

$$\text{Prob}^{\times}\{K = k\} = (1 - F_V(\epsilon))^{k-1} F_V(\epsilon),$$

where $\text{Prob}^{\times}$ denotes the product probability measure over $\omega^{(1)}, \omega^{(2)}, \ldots$ The mean of this geometric distribution is $1/F_V(\epsilon)$, whence

$$\mathbb{E}\{K\} = \frac{1}{F_V(\epsilon)} \leq \frac{1}{1 - \beta_\epsilon(N)},$$

which proves the second point (note that equality holds if the scenario problem is f.s. w.p. one). The cumulative of the above geometric distribution is

$$\text{Prob}^{\times}\{K \leq k\} = 1 - (1 - F_V(\epsilon))^k.$$

This function is increasing in $F_V(\epsilon)$, thus $F_V(\epsilon) \geq 1 - \beta_\epsilon(N)$ implies

$$\text{Prob}^{\times}\{K \leq k\} \geq 1 - \beta_\epsilon(N)^k,$$

which proves the third point. □

B. Proof of Lemma 1

At any given iteration k of Algorithm 2, let us consider the sequence of binary random variables appearing inside the ϵ'-RVO:

$$v_i = \begin{cases} 1 & \text{if } f(\theta_k^*, q^{(i)}) > 0 \\ 0 & \text{otherwise,} \end{cases} \qquad i = 1, \ldots, N_o$$

By definition, we have that $\text{Prob}\{q : f(\theta_k^*, q) > 0\} = V(\theta_k^*)$, and $V(\theta_k^*)$ is a random variable with cumulative distribution function given by F_V. Therefore, for *given* $V(\theta_k^*) = p$, the v_is form an i.i.d. Bernoulli sequence with success probability p. However, p is itself a random variable having cumulative distribution F_V. Therefore, the v_is form a so-called *conditionally i.i.d.* Bernoulli sequence [1], having F_V as the directing de Finetti measure. In simpler terms, the v_is are described by a *compound* distribution: first a success probability p is extracted at random according to its directing distribution F_V, and then the v_is are generated according to an i.i.d. Bernoulli distribution with success probability p. Let $S \doteq \sum_{i=1}^{N_o} v_i$. Conditional on $V(\theta_k^*) = p$, the random variable S has Binomial distribution $\text{Bin}(N_o, p)$ thus, from (6),

$$\begin{aligned} \text{Prob}^{N_o}\{S \leq z | V(\theta_k^*) = p\} &= \sum_{i=0}^{\lfloor z \rfloor} \binom{N_o}{i} p^i (1-p)^{N_o - i} \\ &= \text{Fbeta}(N_o - \lfloor z \rfloor, \lfloor z \rfloor + 1, 1 - p) \\ &= 1 - \text{Fbeta}(\lfloor z \rfloor + 1, N_o - \lfloor z \rfloor, p) \\ &\leq 1 - \text{Fbeta}(z + 1, N_o - z, p). \end{aligned} \tag{23}$$

Considering Eq. (4), we next let

$$F_V(t) \doteq \text{Fbeta}(n, N + 1 - n; t) + \Psi(t), \quad t \in [0, 1], \tag{24}$$

where $\Psi(t)$ is some unknown function such that $0 \leq \Psi(t) \leq 1 - \text{Fbeta}(n, N + 1 - n; t)$, for all $t \in [0, 1]$, and $\Psi(0) = \Psi(1) = 0$. Observe that $\Psi(t)$ is identically zero if the scenario problem is f.s. w.p. one. Consider the event

$$\begin{aligned} \text{GoodTrue} &\doteq \{\text{True} \cap V(\theta_k^*) \leq \epsilon\} \\ &= \{S \leq \lfloor \epsilon' N_o \rfloor \cap V(\theta_k^*) \leq \epsilon\}. \end{aligned}$$

Leting $z \doteq \lfloor \epsilon' N_o \rfloor$, we have that

$$\begin{aligned} \text{Prob}^{N+N_o}\{\text{GoodTrue}\} &= \text{Prob}^{N+N_o}\{S \leq z \cap V(\theta_k^*) \leq \epsilon\} \\ &= \int_0^\epsilon \text{Prob}^{N_o}\{S \leq z | V(\theta_k^*) = t\} \mathrm{d}F_V(t) \\ \text{[using (23)]} &= \int_0^\epsilon \text{Fbeta}(N_o - z, z + 1; 1 - t) \mathrm{d}F_V(t) \\ \text{[using (24)]} &= (1 - H_{\epsilon,\epsilon'}(N, N_o)) + R(\epsilon), \end{aligned} \tag{25}$$

where we defined

$$\begin{aligned} H_{\epsilon,\epsilon'}(N, N_o) &\doteq 1 - \int_0^\epsilon \text{Fbeta}(N_o - z, z + 1; 1 - t) \cdot \\ &\qquad \cdot \text{beta}(n, N + 1 - n; t) \mathrm{d}t \end{aligned} \tag{26}$$

$$R(\epsilon) \doteq \int_0^\epsilon \text{Fbeta}(N_o - z, z + 1; 1 - t) \mathrm{d}\Psi(t). \tag{27}$$

We next analyze the above two terms. For the first term, we have

$$\begin{aligned} &1 - H_{\epsilon,\epsilon'}(N, N_o) \\ &= \int_0^\epsilon \text{Fbeta}(N_o - z, z + 1; 1 - t) \text{beta}(n, N + 1 - n; t) \mathrm{d}t \\ &\text{[using (23)]} = \sum_{i=0}^{z} \int_0^\epsilon \binom{N_o}{i} t^i (1 - t)^{N_o - i} \cdot \\ &\qquad \cdot \text{beta}(n, N + 1 - n; t) \mathrm{d}t \\ &= \sum_{i=0}^{z} \int_0^\epsilon \binom{N_o}{i} \frac{1}{B(n, N + 1 - n)} t^{i+n-1} (1 - t)^{N_o - i + N - n} \mathrm{d}t \\ &= \sum_{i=0}^{z} \int_0^\epsilon \binom{N_o}{i} \frac{B(i + n, N_o - i + N - n + 1)}{B(n, N + 1 - n)} \cdot \\ &\qquad \cdot \text{beta}(i + n, N_o - i + N - n + 1; t) \mathrm{d}t \\ &\text{[by def. in (7)]} = \sum_{i=0}^{z} \int_0^\epsilon f_{\text{bb}}(N_o, n, N + 1 - n; i) \cdot \\ &\qquad \cdot \text{beta}(i + n, N_o - i + N - n + 1; t) \mathrm{d}t \\ &= \sum_{i=0}^{z} f_{\text{bb}}(N_o, n, N + 1 - n; i) \cdot \\ &\qquad \cdot \text{Fbeta}(n + i, N + N_o - n - i + 1; \epsilon). \end{aligned} \tag{28}$$

Observe that, for all $i = 0, \ldots, z$, it holds that

$$\begin{aligned} &\text{Fbeta}(n + i, N + N_o - n - i + 1; \epsilon) \\ &= \sum_{j=n+i}^{N+N_o} \binom{N + N_o}{j} \epsilon^j (1 - \epsilon)^{N + N_o - j} \\ &\geq \sum_{j=n+z}^{N+N_o} \binom{N + N_o}{j} \epsilon^j (1 - \epsilon)^{N + N_o - j} \\ &= \text{Fbeta}(n + z, N + N_o - n - z + 1; \epsilon). \end{aligned}$$

Therefore, we obtain following bound

$$\begin{aligned} &1 - H_{\epsilon,\epsilon'}(N, N_o) \\ &\geq \text{Fbeta}(n + z, N + N_o - n - z + 1; \epsilon) \\ &\qquad \cdot \sum_{i=0}^{z} f_{\text{bb}}(N_o, n, N + 1 - n; i) \\ &= (1 - \text{Fbeta}(N + N_o - n - z + 1, n + z; 1 - \epsilon)) \cdot \\ &\qquad \cdot (1 - H_{1,\epsilon'}(N, N_o)). \end{aligned}$$

For $z = \lfloor \epsilon' N_o \rfloor$, we have, in particular, that

$$\begin{aligned}
&1 - H_{\epsilon,\epsilon'}(N, N_o) \\
&\quad \geq \big(1 - \mathrm{Fbeta}(N + N_o - n - \lfloor \epsilon' N_o \rfloor + 1, n + \lfloor \epsilon' N_o \rfloor; 1 - \epsilon)\big) \\
&\qquad\qquad \cdot (1 - H_{1,\epsilon'}(N, N_o)) \\
&\quad \geq \big(1 - \mathrm{Fbeta}(N + (1 - \epsilon')N_o - n + 1, n + \epsilon' N_o; 1 - \epsilon)\big) \\
&\qquad\qquad \cdot (1 - H_{1,\epsilon'}(N, N_o)) \\
&\quad \doteq \big(1 - \bar{\beta}_{\epsilon,\epsilon'}(N, N_o)\big) \cdot (1 - H_{1,\epsilon'}(N, N_o)) \,. \qquad (29)
\end{aligned}$$

We next consider the $R(\epsilon)$ term in (27). We have that

$$\begin{aligned}
R(\epsilon) &= \int_0^\epsilon (1 - \mathrm{Fbeta}(z + 1, N_o - z; t)) \mathrm{d}\Psi(t) \qquad (30) \\
&= \Psi(\epsilon) - \int_0^\epsilon \mathrm{Fbeta}(z + 1, N_o - z; t) \mathrm{d}\Psi(t) \\
&\text{[integrating by parts]} = \Psi(\epsilon)\mathrm{Fbeta}(N_o - z, z + 1; 1 - \epsilon) \\
&\qquad + \int_0^\epsilon \Psi(t)\mathrm{beta}(z + 1, N_o - z; t)\mathrm{d}t.
\end{aligned}$$

Since $\Psi(t) \geq 0$ forall $t \in [0, 1]$, the above expression shows that $R(\epsilon) \geq 0$ for all $\epsilon \in [0, 1]$, with $R(\epsilon)$ being identically zero for problems that are f.s. Considering (25), this fact permits us to conclude that

$$\begin{aligned}
\mathrm{Prob}^{N+N_o}\{\mathrm{GoodTrue}\} &= \mathrm{Prob}^{N+N_o}\{S \leq z \cap V(\theta_k^*) \leq \epsilon\} \\
&= 1 - H_{\epsilon,\epsilon'}(N, N_o) + R(\epsilon) \\
&\geq 1 - H_{\epsilon,\epsilon'}(N, N_o) \\
\text{[from (29)]} \quad &\geq \big(1 - \bar{\beta}_\epsilon(N, N_o)\big) \cdot \\
&\quad \cdot (1 - H_{1,\epsilon'}(N, N_o)) \,,
\end{aligned}$$

which proves (10) and (11). Also, we obtain that

$$\begin{aligned}
\mathrm{Prob}^{N+N_o}\{\mathrm{True}\} &\doteq \mathrm{Prob}^{N+N_o}\{S \leq z\} \\
&= \mathrm{Prob}^{N+N_o}\{S \leq z \cap V(\theta_k^*) \leq 1\} \\
&= 1 - H_{1,\epsilon'}(N, N_o) + R(1) \\
&\geq 1 - H_{1,\epsilon'}(N, N_o),
\end{aligned}$$

which proves (9). Further, using (25), we have that

$$\begin{aligned}
\mathrm{Prob}^{N+N_o}\{\mathrm{BadTrue}\} &\doteq \mathrm{Prob}^{N+N_o}\{S \leq z \cap V(\theta_k^*) > \epsilon\} \\
&= \mathrm{Prob}^{N+N_o}\{\mathrm{True}\} \\
&\quad - \mathrm{Prob}^{N+N_o}\{\mathrm{GoodTrue}\} \\
&= H_{\epsilon,\epsilon'}(N, N_o) - H_{1,\epsilon'}(N, N_o) \\
&\quad + (R(1) - R(\epsilon)).
\end{aligned}$$

If the scenario problem is f.s., then $R(1) = R(\epsilon) = 0$, hence

$$\begin{aligned}
\mathrm{Prob}^{N+N_o}\{\mathrm{BadTrue}\} &= H_{\epsilon,\epsilon'}(N, N_o) - H_{1,\epsilon'}(N, N_o) \\
&\quad + (R(1) - R(\epsilon)) \\
\text{[if problem is f. s.]} \quad &= H_{\epsilon,\epsilon'}(N, N_o) - H_{1,\epsilon'}(N, N_o) \\
\text{[using (29)]} \quad &\leq \bar{\beta}_{\epsilon,\epsilon'}(N, N_o)(1 - H_{1,\epsilon'}(N, N_o)).
\end{aligned}$$

All the above proves (13). To upper bound the probability of BadTrue in the non-fully supported case, we reason instead as follows:

$$\begin{aligned}
&\mathrm{Prob}^{N+N_o}\{\mathrm{BadTrue}\} \\
&\quad = \int_\epsilon^1 \mathrm{Prob}^{N_o}\{S \leq z | V(\theta_k^*) = t\} \mathrm{d}F_V(t) \\
&\quad = \int_\epsilon^1 \mathrm{Fbeta}(N_o - z, z + 1; 1 - t) \mathrm{d}F_V(t) \\
&\quad \text{[integrand is decreasing in } t] \\
&\quad \leq \int_\epsilon^1 \mathrm{Fbeta}(N_o - z, z + 1; 1 - \epsilon) \mathrm{d}F_V(t) \\
&\quad = \mathrm{Fbeta}(N_o - z, z + 1; 1 - \epsilon) \int_\epsilon^1 \mathrm{d}F_V(t) \\
&\quad = \mathrm{Fbeta}(N_o - z, z + 1; 1 - \epsilon) \cdot \mathrm{Prob}^N\{V(\theta_k^*) > \epsilon\} \\
&\quad \text{[from (4)]} \\
&\quad \leq \mathrm{Fbeta}(N_o - z, z + 1; 1 - \epsilon) \cdot \mathrm{Fbeta}(N + 1 - n, n; 1 - \epsilon) \\
&\quad \text{[since } z = \lfloor \epsilon' N_o \rfloor] \\
&\quad \leq \mathrm{Fbeta}((1 - \epsilon')N_o, \epsilon' N_o + 1; 1 - \epsilon) \cdot \beta_\epsilon(N),
\end{aligned}$$

which proves (13). □

C. Proof of Theorem 3

Let us define the event $\mathrm{BadExit}_k$ as the one where the algorithm reaches the k-th iteration, and then exits with a "bad" solution, i.e., with a solution θ_k^* for which $V(\theta_k^*) > \epsilon$. The probability of this event is the probability that the ϵ'-RVO returns `false` precisely $k - 1$ times (for this guarantees that we reach the k-th iteration), and then the event BadTrue happens at the k-th iteration. Therefore, letting q denote the probability of BadTrue, and p denote the probability of True (events defined as in Lemma 1) we have that

$$\mathrm{Prob}^{\times\times}\{\mathrm{BadExit}_k\} = (1 - p)^{k-1} q.$$

The event BadExit in which the algorithm terminates with a bad solution is the union of the non-overlapping events $\mathrm{BadExit}_k$, $k = 1, 2, \ldots$, therefore

$$\begin{aligned}
\mathrm{Prob}^{\times\times}\{\mathrm{BadExit}\} &= \sum_{k=1}^\infty \mathrm{Prob}^{\times\times}\{\mathrm{BadExit}_k\} \\
&= \sum_{k=1}^\infty (1 - p)^{k-1} q = q \sum_{k=0}^\infty (1 - p)^k \\
&= \frac{q}{p} = \frac{\mathrm{Prob}^{N+N_o}\{\mathrm{BadTrue}\}}{\mathrm{Prob}^{N+N_o}\{\mathrm{True}\}}.
\end{aligned}$$

We now use (12) to upper bound q, and then use (9) to conclude that

$$\begin{aligned}
&\mathrm{Prob}^{\times\times}\{\mathrm{BadExit}\} \\
&\quad \leq \frac{\mathrm{Fbeta}((1 - \epsilon')N_o, \epsilon' N_o + 1; 1 - \epsilon)\beta_\epsilon(N)}{1 - H_{1,\epsilon'}(N, N_o)},
\end{aligned}$$

which proves (14). In the fully supported case, we can instead use (13) to upper bound q, and hence conclude that

$$\mathrm{Prob}^{\times\times}\{\mathrm{BadExit}\} \leq \bar{\beta}_{\epsilon,\epsilon'}(N, N_o),$$

which proves (15).

Let next K denote the running time of Algorithm 2, that is the value of the iteration count when the algorithm terminates. Since the algorithm terminates as soon as a True event happens, and since the True events are statistically independent among iterations, we

have that $\{K = k\}$ has geometric probability $(1-p)^{k-1}p$, where p is the probability of True. Therefore, the expected value of K is $1/p \leq 1/(1 - H_{1,\epsilon'}(N, N_o))$, where the inequality follows from (9), and this proves point 2 in the theorem. Via the same reasoning, $\{K > k\}$ has probability $(1-p)^k$, and hence we conclude that

$$\text{Prob}^{\times\times}\{K \leq k\} = 1 - (1-p)^k \geq 1 - H_{1,\epsilon'}(N, N_o)^k,$$

which proves the third point in the theorem. □

D. Proof of Corollary 1

From from Eq. (28) we have that, for $z \doteq \lfloor \epsilon' N_o \rfloor$,

$$1 - H_{1,\epsilon'}(N, N_o) = \int_0^1 \text{Fbeta}(N_o - z, z+1; 1-t)\text{beta}(n, N+1-n; t)\text{d}t. \tag{31}$$

We recall that a beta(α, β) density has mean $\alpha/(\alpha+\beta)$, peak (mode) at $(\alpha-1)/(\alpha+\beta-2)$, and variance $\sigma^2 = \alpha\beta/((\alpha+\beta)^2(\alpha+\beta+1))$. Then, we observe that $\text{Fbeta}(N_o - z, z+1; 1-t) = 1 - \text{Fbeta}(z+1, N_o - z; t)$, where $\text{Fbeta}(z+1, N_o - z; t)$ is the cumulative distribution of a beta$(z+1, N_o - z)$ density. The peak of this density is at $z/(N_o - 1)$, which tends to ϵ' for $N_o \to \infty$; further, the variance of this distribution goes to zero as $O(N_o^{-1})$, which permits us to argue that, for large N_o, the function $\text{Fbeta}(N_o - z, z+1; 1-t)$ has an inflection point near ϵ' and decreases rapidly from value $\simeq 1$ to value $\simeq 0$ as t crosses ϵ'. That is, as $N_o \to \infty$, the function $\text{Fbeta}(N_o - z, z+1; 1-t)$ tends to a step function which is one for $t < \epsilon'$ and zero for $t > \epsilon'$. Therefore, we have for the integral in (31) that

$$\begin{aligned} 1 - H_{1,\epsilon'}(N, N_o) &\to \int_0^{\epsilon'} 1 \cdot \text{beta}(n, N+1-n; t)\text{d}t \\ &= \text{Fbeta}(n, N+1-n; \epsilon') = 1 - \beta_{\epsilon'}(N), \end{aligned}$$

which proves (16). □

References

[1] D. Aldous. Exchangeability and related topics. In P.L. Hennequin, editor, *École d'Été de Probabilités de Saint-Flour XIII*, volume 1117 of *Lecture Notes in Math.*, pages 1–198. Springer, 1983.

[2] G.C. Calafiore. Random convex programs. *SIAM Journal on Optimization*, 20(6):3427–3464, 2010.

[3] G.C. Calafiore and M.C. Campi. The scenario approach to robust control design. *IEEE Transactions on Automatic Control*, 51(5):742–753, 2006.

[4] G.C. Calafiore and F. Dabbene. A probabilistic analytic center cutting plane method for feasibility of uncertain LMIs. *Automatica*, 43:2022–2033, 2007.

[5] G.C. Calafiore, F. Dabbene, and R. Tempo. Research on probabilistic methods for control system design. *Automatica*, 47:1279–1293, 2011.

[6] G.C. Calafiore and L. El Ghaoui. *Optimization Models*. Cambridge University Press, 2014.

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

[8] M. Chamanbaz, F. Dabbene, R. Tempo, V. Venkataramanan, and Q. G. Wang. Sequential randomized algorithms for convex optimization in the presence of uncertainty. *IEEE Transactions on Automatic Control*, 2015; 10.1109/TAC.2015.2494875.

[9] S. Garatti and M.C. Campi. Modulating robustness in control design: principles and algorithms. *IEEE Control Systems Magazine*, 33:36–51, 2013.

[10] Michael Grant and Stephen Boyd. CVX: Matlab software for disciplined convex programming, version 2.1. http://cvxr.com/cvx, March 2014.

[11] J.C. Lee and D.J. Sabavala. Bayesian estimation and prediction for the beta-binomial model. *Journal of Business and Economic Statistics*, 5(3):357–367, 1987.

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

[13] A. Nemirovski and A. Shapiro. Convex approximations of chance constrained programs. *SIAM Journal on Optimization*, 17(4):969–996, 2009.

[14] B.K. Pagnoncelli, S. Ahmed, and A. Shapiro. Sample average approximation method for chance constrained programming: Theory and applications. *J. Optim. Theory Appl.*, 142:399–416, 2009.

[15] I. Petersen and R. Tempo. Robust control of uncertain systems: Classical results and recent developments. *Automatica*, 50:1315–1335, 2014.

[16] A. Prékopa. Probabilistic programming. In A. Rusczyński and A. Shapiro, editors, *Stochastic Programming*, volume 10 of *Handbooks in Operations Research and Management Science*. Elsevier, Amsterdam, 2003.

[17] A. Rantzer. Distributed control of positive system. arXiv:1203.0047v3, 2014.

[18] R. Tempo, G.C. Calafiore, and F. Dabbene. *Randomized Algorithms for Analysis and Control of Uncertain Systems: With Applications*. Communications and Control Engineering. Springer-Verlag; 2nd Ed., 2013.

[19] Eric W. Weisstein. Beta binomial distribution. MathWorld–A Wolfram Web Resource.

G.C. Calafiore received the "Laurea" degree in Electrical Engineering from Politecnico di Torino in 1993, and the Ph.D. degree in Information and System Theory from Politecnico di Torino, in 1997. Since 1998 he was with the faculty of Dipartimento di Automatica e Informatica, Politecnico di Torino, where he currently serves as a full professor. Dr. Calafiore held several visiting positions at international institutions: at the Information Systems Laboratory (ISL), Stanford University, California, in 1995; at the Ecole Nationale Supérieure de Techniques Avanceés (ENSTA), Paris, in 1998; and at the University of California at Berkeley, in 1999, 2003 and 2007. He had an appointment as a Senior Fellow at the Institute of Pure and Applied Mathematics (IPAM), University of California at Los Angeles, in 2010. He has been an Associate Editor for the IEEE Transactions on Systems, Man, and Cybernetics (T-SMC), for the IEEE Transactions on Automation Science and Engineering (T-ASE), and for Journal Européen des Systémes Automatisés (JESA). Dr. Calafiore is the author of more than 160 journal and conference proceedings papers, and of eight books. He is a senior member of the IEEE. He received the IEEE Control System Society "George S. Axelby" Outstanding Paper Award in 2008. His research interests are in the fields of convex optimization, randomized algorithms, identification, and control of uncertain systems, with applications ranging from finance and economic systems to robust control, machine learning, and robotics.